ISSN 0265-2919

80p 47

THE HOME COMPUTER ADVANCED COURSE

MAKING THE MOST OF YOUR MICRO

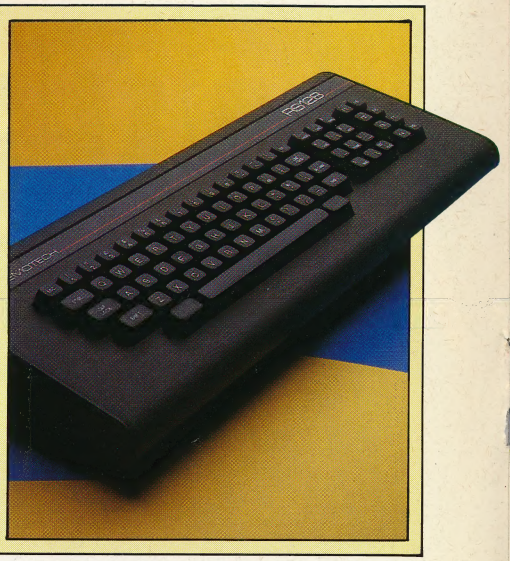
An ORBIS Publication

IR £1 Aus $1.95 NZ $2.25 SA R1.95 Sing $4.50 USA & Can $1.95

CONTENTS

Next Week

- **Having now constructed our robot, we begin to look at some practical applications programs.**
- **Memotech have recently upgraded their MTX 500 series of computers. We examine the improvements in the RTS128.**
- **We review two books that speculate how computers will change the shape of society.**



QUIZ

1) Where would you find a 'software board'?
2) Who led the team in the birth of the Eagle?
3) Why does data have to be 'detokenised' before it can be transmitted?

Answers To Last Week's Quiz

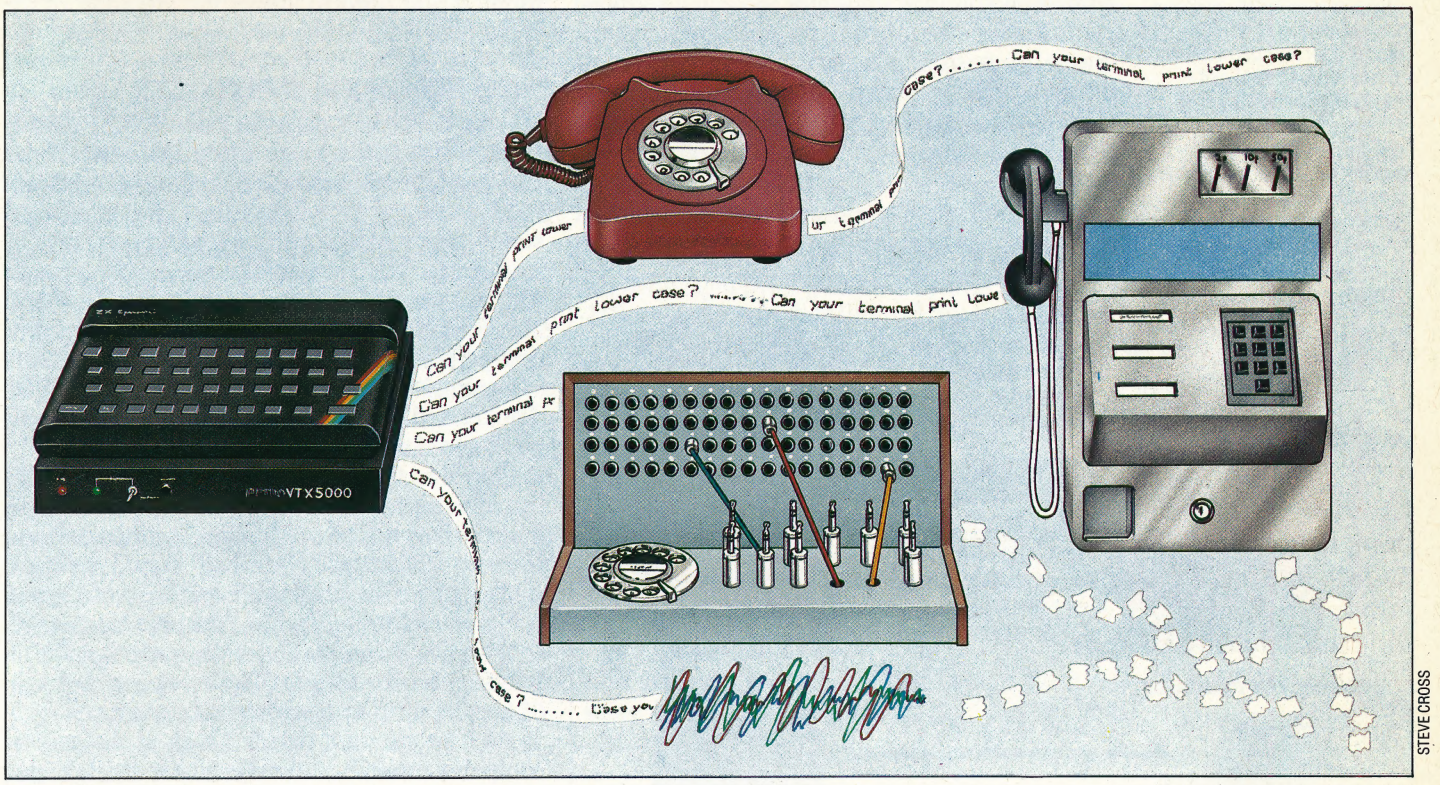
1) An acoustic modem — also known as an acoustic coupler — requires a telephone handset.
2) The Canon T70 chip contains 16 bytes of RAM.
3) Parity checking is used in computer communications to ensure that the correct signal has been received.
4) In LOGO, 'state transparent' means that after a procedure has been executed, the turtle is pointing in the same direction as it was before the procedure commenced.

Editor Mike Wesley; **Art Editor** Claudia Zeff; **Technical Editor** Brian Morris; **Production Editor** Robert Pickering; **Designer** Julian Dorr; **Art Assistant** Liz Dixon; **Staff Writer** Stephen Malone; **Consultant Editor** Steve Colwill; **Contributors** Geoff Bains, Harvey Mellar, Jim Lennox, Surya, Steve Colwill, Andrew Bangham; **Software Consultants** Pilot Software City; **Group Art Director** Perry Neville; **Managing Director** Stephen England; **Published by Orbis Publishing Ltd: Editorial Director** Brian Innes; **Project Development** Peter Brookesmith; **Executive Editor** Maurice Geller; **Production Controller** Peter Taylor-Medhurst; **Subscription Manager** Christine Allen; **Designed and produced by Bunch Partworks Ltd; Editorial Office** 14 Rathbone Place, London W1P 1DE; **© APSIF Copenhagen 1984; © Orbis Publishing Ltd 1984: Typeset by Universe; Reproduction by Mullis Morgan Ltd; Printed in Great Britain by Artisan Press Ltd, Leicester**

HOW TO OBTAIN ISSUES AND BINDERS FOR THE HOME COMPUTER ADVANCED COURSE – Issues can be obtained by placing an order with your newsagent or direct from our subscription department. If you have any difficulty obtaining any back issues from your newsagent, please write to us stating the issue(s) required and enclosing a cheque for the cover price of the issue(s). **AUSTRALIA** – please write to: Gordon & Gotch (Aus) Ltd, 114 William Street, PO Box 767G, Melbourne, Victoria 3001. **MALTA, NEW ZEALAND & SOUTH AFRICA** – Back numbers are available at cover price from your newsagent. In case of difficulty, write to the address given for binders.
UK/EIRE – Price: 80p/IR£1. Subscription: 6 months: £23.92. 1 Year: £47.84. Binder: please send £3.95 per binder, or take advantage of our special offer in early issues. **EUROPE** – Price: 80p. Subscription: 6 months air: £43.68. Surface: £34.84. 1 year air: £87.36. Surface: £69.68. Binder: £5.00. Airmail: £8.25. **MALTA** – Obtain binders from your newsagent or Miller (Malta) Ltd, MA Vassalli Street, Valetta, Malta. Price: £3.95. **MIDDLE EAST** – Price: 80p. Subscription: 6 months air: £45.76. Surface: £34.84. 1 year air: £91.52. Surface: £69.68. Binder: £5.00. Airmail: £8.25. **AMERICAS/ASIA/AFRICA** – Price: US/CAN$1.95/80p. Subscription: 6 months air: £54.08. Surface: £34.84. 1 year air: £108.16. Surface: £69.68. Binder: £5.00. Airmail: £9.50. **SOUTH AFRICA** – Price: SA R1.95. Obtain binders from any branch of Central News Agency or Intermag, PO Box 57394, Springfield 2137. **SINGAPORE** – Price: Sing $4.50. Obtain binders from MPH Distributors, 601 Sims Drive, 03-07-21, Singapore 1438. **AUSTRALASIA/FAR EAST** – Price: 80p. Subscription: 6 months air: £58.24. Surface: £34.84. 1 year air £116.48. Surface: £69.68. Binder: £5.00. Airmail: £9.75. **AUSTRALIA** – Price: Aus$1.95. Obtain binders from First Post Pty Ltd, 23 Chandos Street, St Leonards, NSW 2065. **NEW ZEALAND** – Price: NZ$2.25. Obtain binders from your newsagent or Gordon & Gotch (NZ) Ltd, PO Box 1595, Wellington.
ADDRESS FOR BINDERS AND BACK ISSUES – Orbis Publishing Limited, Orbis House, Bedfordbury, London WC2 4BT. Telephone 01-379 6711. Cheques/postal orders should be made payable to Orbis Publishing Limited. Binder prices include postage and packing and prices are in stirling. Back issues are sold at the cover price, and we do not charge carriage in the UK.
NOTE – Binders and back issues are obtainable subject to availability of stocks. Whilst every attempt is made to keep the price of the issues and binders constant, the publishers reserve the right to increase the stated prices at any time when circumstances dictate. Binders depicted in this publication are those produced for the UK and Australian markets only. Binders and Issues may be subject to import duty and/or local taxes, which are not included in the above prices unless stated.
ADDRESS FOR SUBSCRIPTIONS – Orbis Publishing Limited, Hurst Farm, Baydon Road, Lambourn Woodlands, Newbury Berks, RG16 7TW. Telephone: 0488-72666. All cheques/postal orders should be made payable to Orbis Publishing Limited, Postage and packaging is included in subscription rates, and prices are given in sterling.

COVER PHOTOGRAPHY BY JAY MURDLE

HOW DO YOU DO?



STEVE CROSS

Out Of Place
Advertisements often show portable micros and battery-powered acoustic modems being used in all sorts of unlikely places, but there are some situations where you *can't* use them. A few examples are ordinary payphones, manually-operated switchboards (used in many hotels) and on static-filled lines (a problem in many remote areas)

We look at further aspects of the operation of a communications system involving a home micro and a modem. In particular, we explain what is meant by the term 'duplex', and discuss the protocols necessary before two home computers can communicate information between one another.

The term 'duplex' tells us whether or not a system can transmit and receive data at the same time. A full-duplex system allows bi-directional data transfer, while half-duplex systems do not. This can be likened to the difference between talking on a telephone, where both parties can talk at the same time, and using a CB radio, where pressing the transmit button automatically prevents you from receiving.

The main advantage of a full-duplex system is its ability to interrupt the transmitting computer. If, for example, it is transmitting a long menu and you already know that you want option 1, you can press '1' and the system will immediately act on that input. In a half-duplex system, you would have to wait until the menu had been completely transmitted before you made your selection.

Full duplex is obtained by using different frequencies for each computer. The computer making the call uses a frequency known as the 'originate', while the host computer uses the 'answer' frequency.

TERMINAL PROTOCOLS

When a computer transmits data to another over a telephone line, it initially knows nothing about the computer at the receiving end. The terminal could be a Spectrum with a 32-column display or a 132-column teletype. It may or may not have colour. It may or may not support lower case characters. In short, it could be anything from a micro to a mainframe.

There are three ways of dealing with this problem. The first is the approach adopted by the BASICODE language (see page 241), which assumes the lowest common denominator and transmits only those formats that the most poorly-equipped terminal can handle. The second is to ask for information about the capabilities of the particular terminal and then modify the output to suit it. The other approach is where the system assumes that it has accessed a particular terminal, or range of terminals, and leaves the user's software to cope with the transmitted data.

The first approach is rarely used, since the lowest common denominator among terminals is the teletype standard. This has an upper-case character set only, no print formatting, no colour, no graphics and a slow speed. The second

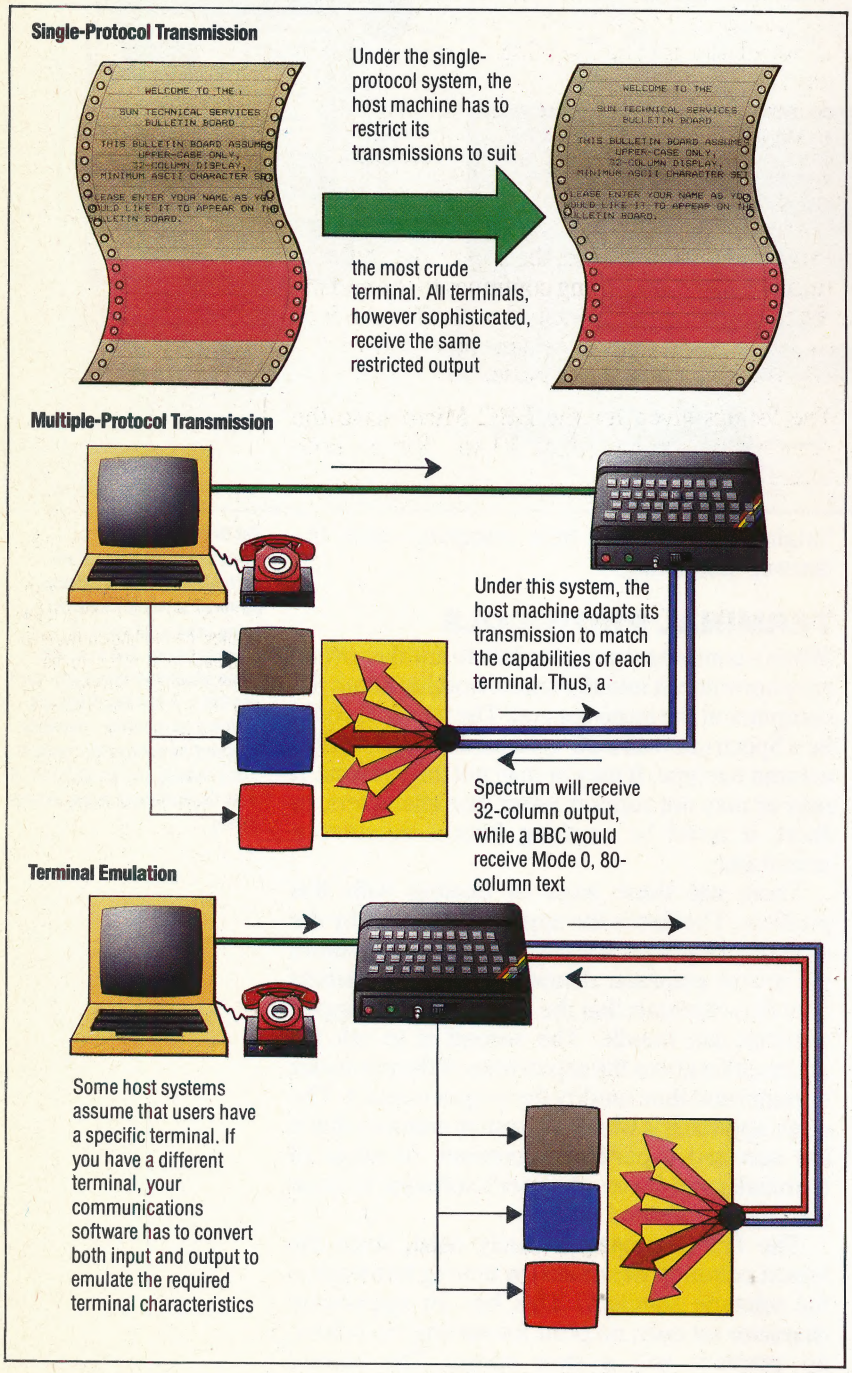
Observing Protocol

There are three main ways to approach communications protocols. The first is to assume the lowest common denominator (slow baud rate, narrow column-width, no colour or graphics, and so on). The second is for the host machine to modify its output to suit a variety of terminals (the technique used by most bulletin boards). The third is for the terminal to 'act like' a particular terminal — this is achieved through a software technique known as 'terminal emulation'

approach is more common, and is used by most bulletin boards. The third method is generally used by commercial and university systems, and relies on a software technique known as 'terminal emulation'.

As the name suggests, terminal emulation is simply a method of persuading a micro to act like a particular terminal. Very simply, terminal emulation software translates incoming terminal control characters (such as those used to clear the screen or position the cursor) into commands understood by the computer being used. Similarly, if the host expects a control character sequence from the terminal, the emulation software will supply it.

Almost all systems will understand a subset of ASCII control characters (for the full ASCII set see page 77). Some of the more useful characters are Control-S (ASCII 19, known as 'XOFF'), which temporarily halts data reception; Control-Q (ASCII 17, known as 'XON'), which restarts it; Control-J (ASCII 10, known as 'LF'), which forces a linefeed without a carriage return; and Control-G (ASCII 7, known as 'BEL'), which sounds the console bell. The last character can be useful if you need to attract the attention of the system operator at the receiving terminal.



KEVIN JONES

THE XMODEM PROTOCOL

Having looked in detail at how ASCII data is transmitted, let's now consider how software is communicated. Transmitting software written in BASIC is straightforward. Most micros support some means of detokenising programs (that is, converting them from their compressed program format into ASCII form): on the BBC, we use *SPOOL; on the Commodore 64, we LIST to a disk or tape file; on the Tandy, the instruction CSAVE <filename>,A is used; and so on. The detokenised file is then transmitted and retokenised at the other end.

However, CP/M command (.COM) files cannot be turned into ASCII form, and any attempt to transmit one results in garbage. For this reason, a protocol known as 'XModem' was developed. XModem, which is a feature of some communications software, simply reads in a C/PM file from disk and transmits it in a standard binary format. The receiving terminal, of course, must also support XModem.

We'll be taking a detailed look at the uses of computer communications in a later article in this series. For now, let's take a brief overview of the range of activities a modem makes possible.

Electronic mail is the name given to systems where users can exchange private messages by transmitting them to a mainframe computer, where they are stored until the recipient logs on and retrieves them.

Most home micro users are interested in two particular applications. The first is to exchange software over the telephone. As we've already seen, transmitting BASIC programs is a simple matter, and is quicker, easier and cheaper than swapping cassettes by post. If you're having difficulty with a program you're writing, you can transmit a copy to friends and see if they can help. If they can, they can transmit a working version back to you!

The second hobbyist use is bulletin boards. These allow you to pass on information to other users, leave requests for technical help, swap jokes, download public domain software, play adventure games, and so on. Bulletin boards are run by enthusiasts, and there is normally no charge (although some boards may charge a once-only nominal fee — a pound or so — in order to cover running costs). We'll examine bulletin boards in detail in a later article. In the next instalment, however, we'll look at how to select modems and communications software.

MANY MOTORS

In the previous instalment of Workshop, we showed you how to connect a servo motor to a user port via the buffering system developed earlier in the course. There are eight data lines on a user port, however, and each of these could be connected to a servo motor. Here, we look at how to control several servo motors simultaneously.

There are eight data lines that may be connected to motors, although only four can be used by the buffer box that we have designed. All eight lines could be used by duplicating the output box circuitry for the other four lines.

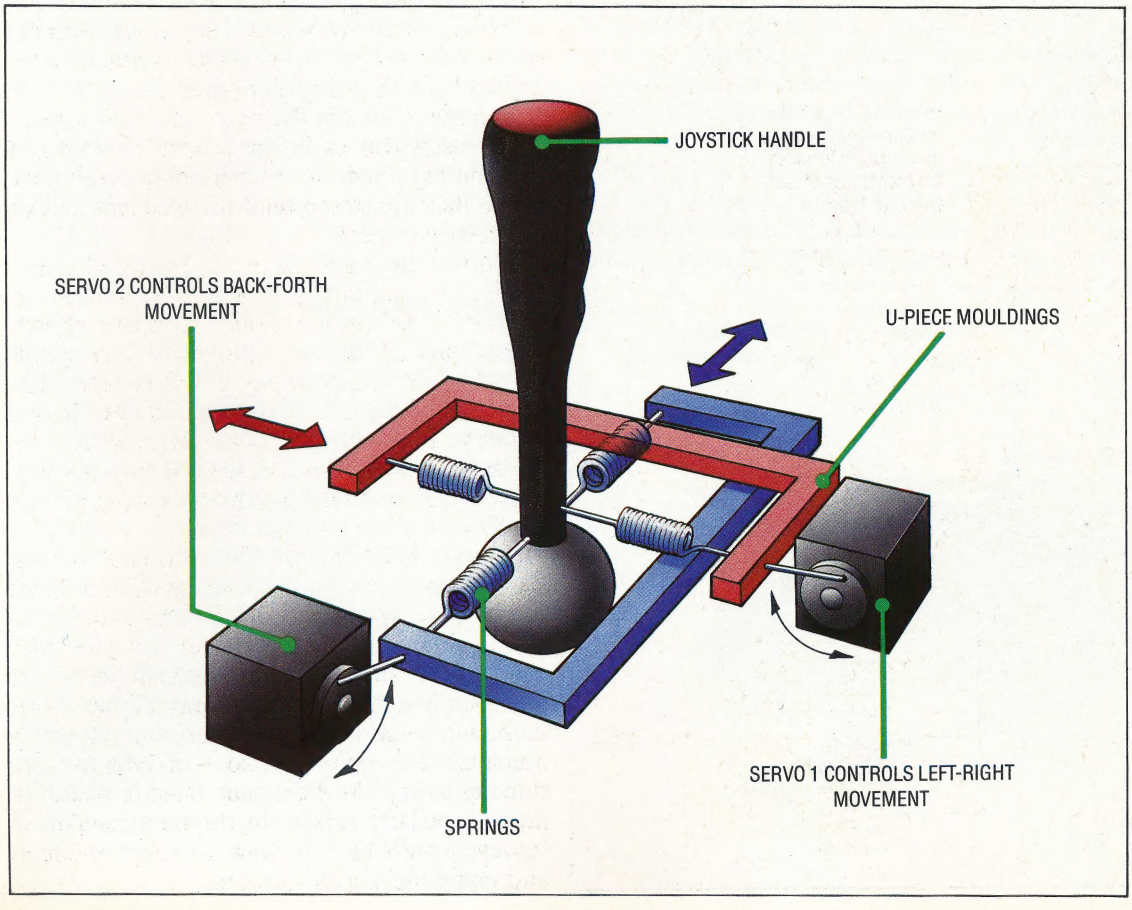
To control eight motors simultaneously, the algorithm for a single motor that we developed on page 894 must be subtly altered. The pulses are all started together, but the second wait loop is replaced by a look-up table. 255 memory locations are reserved for the table and are all set initially to 255 ($FF).

The exceptions — when a motor is switched off — are then entered into this table. For example, if data line 2 is to be turned off after a count of 20, the 20th entry in the table will be altered from binary 11111111 ($FF) to binary 11111011 ($FB). Notice that in the Assembly listing, this is done by ANDing with the value already in the table. Once all eight exceptions have been entered in the table, the wait loop is started, but this time each element of the table is ANDed to the user port.

The algorithm for multiple motor control is:

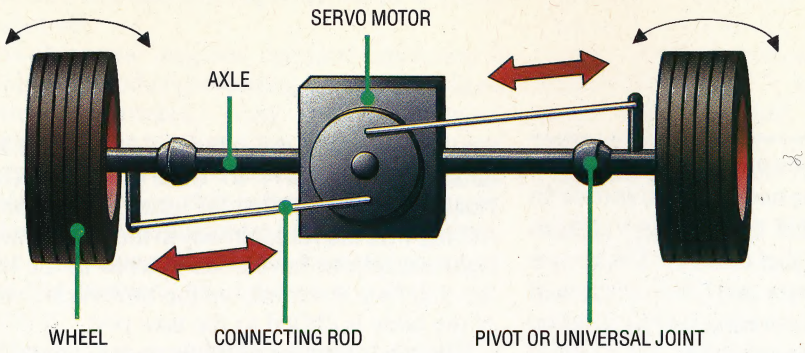
1) Specify the angle of each motor by storing the angles in eight bytes (ANGLE+0 to ANGLE+7).
2) Set all user port data bits high to start the pulses together.
3) Insert the exceptions into the look-up table.
4) Wait for one millisecond.
5) Load the accumulator with binary 11111111 ($FF). Then AND the accumulator with each element of the look-up table in turn. When an exception is encountered, the appropriate bit will turn off. Since the ANDing continues to the end of the table, this bit will remain turned off to the end.
6) Set the exceptions in the table back to binary 11111111, ready for the next pulse.

The listings given for the BBC Micro have the same initial routine (lines 10 to 280) in both



Joystick Feedback
Joysticks are normally used to provide directional information to be used by software. A possible computer application for servo motors is to use them to allow controlling software to feed back information to the joystick handle. Two servos are used to push and pull the handle in each horizontal plane, providing important feedback data in sophisticated flight simulators. A joystick that 'kicks' under the hand in response to a control movement by the pilot improves the simulation by providing tactile, as well as visual, information to the user

Steering Mechanism



In our Workshop robot, directional control is achieved by independent control of two bi-directional stepper motors. A possible alternative to this arrangement is to have one stepper motor driving the vehicle, and a servo motor to steer the vehicle, in much the same way as a driver steers a car. The diagram shows a servo motor mounted directly over the wheel axle, pushing or pulling connecting rods to the wheels to steer the vehicle. An alternative arrangement would be to use a rack-and-pinion steering mechanism, mounting the pinion gear on the servo motor's spindle

programs. The first listing is for control of a single servo motor connected to one of the user port lines. The event timer is set up using BASIC. An initialising procedure assembles the event-handling routine before the main program runs it by enabling event 5.

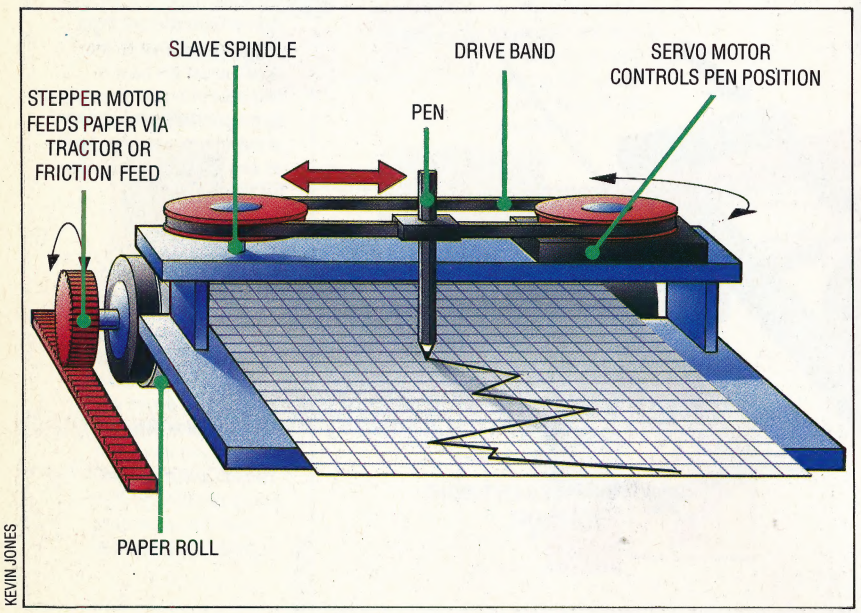
On the BBC Micro, the operating system includes a centisecond user timer. By setting it to 2 centiseconds (two interrupts of 10 milliseconds each) and then using the 'event' vector (see page 464 of the BBC Advanced User Guide), the processor will jump to the pulser code at the correct time. Since the operating system was designed to use the events, the program needs merely to return (RTS) from the subroutine, and not use RTI.

The second listing, for multiple servo control, first uses a BASIC loop to initialise the look-up table with $FF values. If each element of the table was output to the user port in turn, the pulses would all continue for two milliseconds. However, exceptions can be made, and each motor turned off in turn. The exceptions are inserted into the table starting with motor 7, by using an offset (in the X register) proportional to the length of the pulse. Then the table is output to the user port by addressing each element in turn indirectly, this time using post-indexed indirect addressing (where the processor adds the value of the Y register to the address found in a zero page vector).

The bit patterns (exceptions) that must be sent to the user port to control the motors are produced as follows. Binary 01111111 is required to turn off motor 7; likewise 10111111 for motor 6; 11011111 for motor 5; and so forth. These are generated by loading the A register with $FF and clearing the carry flag. Then, as the routine handles each exception in turn, these bits are rotated once to the right. On the first rotation, the carry bit is moved to bit 7, bit 7 moves to bit 6, and so on, with bit 0 replacing the carry bit. The bit pattern required to turn off each motor is temporarily saved on the stack. Each table entry is ANDed with the previous one (as it is output) to ensure that once a pulse is turned off, it remains off.

Having generated the machine code, event 5 is cleared for action. Thereafter, pressing the Shift key along with a number key from 1 to 8 selects one of the motors, while pressing keys 1 to 9 moves the motors into position.

Chart Recorder
A chart recorder can be designed using a servo motor to move the pen back and forth, while a stepper motor feeds the paper underneath the pen. Angular movement of the servo motor spindle is translated into the linear back-and-forth movement of the pen by way of a taut drive band. The pen movement corresponds to changes in a vertical axis variable — for example, temperature or barometric pressure — with the continuous paper feed often corresponding to the passage of time



KEVIN JONES

USING THE LISTINGS

To use the BBC listings, simply type them in, as shown, SAVE them, and then RUN. Both the Single Servo and Multiple Servo listings are run with the Common Initial Routine (lines 10 to 750).

For the Commodore 64, the second algorithm uses the same command keys as the BBC listing. The BASIC Calling program allows the position of each motor to be set independently: the Shift key and a number key from 1 to 8 select the desired motor, and a key from 1 to 9 defines the required position.

If you have an assembler, then type in the source listing and assemble it into an object file that can be subsequently loaded by the BASIC Calling program. Alternatively, type in the BASIC Loader for the machine code and RUN this. Type NEW before loading and running the BASIC Calling program. If you use the BASIC Loader then lines 30 and 40 can be omitted.

Commodore 64 Multiple Servo Control

Source Code

```
1000 ;******************
1010 ;******************
1020 ;++                ++
1030 ;++ CBM MULTIPLE   ++
1040 ;++ SERVO CONTROL  ++
1050 ;++                ++
1060 ;******************
1070 ;******************
1080 ;
1090 PORT = 56577 ;USER PORT DATA REGISTER
1100 ANGLE=12288  ;ANGLE VALUE LOCATION
1110 ZPAGE=$FB    ;0 PAGE POINTER TO TABLE
1120 ;
1130 *=$0334
1140 ;
1150        SEI         ;INTERRUPTS OFF
1160        LDA $0314 ;EXISTING IRQ VECTOR
1170        LDX $03C4
1180        STA $03C4
1190        STX $0314
1200        LDA $0315
1210        LDX $03C5
1220        STA $03C5
1230        STX $0315
1240 ;
1250 ;++++ INITIALISE TABLE ++++
1260 ;
1270        LDA #$FF
1280        LDY #$00
1290 TABLE
1300        STA (ZPAGE),Y
1310        DEY
1320        BNE TABLE
1330        CLI         ;INTERRUPTS ON
1340        RTS
1350 ;
1360 ;++++ EVENT HANDLER ++++
1370 ;
1380        PHP
1390        PHA         ;SAVE REGISTERS
1400        TYA         ;ON STACK
1410        PHA
1420        TXA
1430        PHA
1440 ;
1450 ;++ START PULSE, FOR SOME MOTORS IT
1460 ;MAY BE POSSIBLE TO START BEFORE FILLING
1470 ;TABLE AND SO REDUCE WAIT LOOP BELOW ++
1480 ;
1490        LDA #$FF
1500        STA PORT
1510 ;++ FILL TABLE WITH EXCEPTIONS ++
1520        LDX #$07
1530        LDA #$FF
1540        CLC
1550 EXCEPT
1560        ROR A
1570        PHA         ;BIT PATTERN
1580        LDY ANGLE,X ;GET MOTOR X OFFSET
1590        AND (ZPAGE),Y ;KEEP EXISTING PATTERN
1600        STA (ZPAGE),Y ;BUT MODIFIED FOR MOTOR X
1610        PLA
1620        DEX
1630        BPL EXCEPT
1640 ;++ TABLE IS NOW LOADED ++
1650        LDY #$30
1660 WAIT
1670        DEY             ;FILL IN SOME
1680        BNE WAIT        ;TIME
1690 ;
1700        LDA #$FF        ;ALL PULSES ON
1710        LDY #$00
1720 LOOP
1730        AND (ZPAGE),Y ;BUT MASK OFF WITH EACH
1740        STA PORT        ;TABLE ELEMENT IN TURN
1750        INY
1760        BNE LOOP
1770 ;
1780        LDX #$07
1790        LDA #$FF
1800 CLEAR
1810        LDY ANGLE,X     ;CLEAR ALL EXCEPTIONS
1820        STA (ZPAGE),Y
1830        DEX
1840        BPL CLEAR
1850 ;++ ALL PULSES SHOULD NOW BE FINISHED ++
1860        PLA
1870        TAX
1880        PLA             ;RESTORE REGISTERS
1890        TAY
1900        PLA
1910        PLP
1920        JMP $EA31
```

BASIC Loader Program

```
10 REM **** BASIC LOADER FOR ***
*
20 REM **** MULTIPLE SERVOS  ***
*
30 :
40 FOR I=820 TO 922
50 READ A:POKE I,A
60 CC=CC+A
70 NEXT I
80 READ CS:IF CS<>CC THEN PRINT"
CHECKSUM ERROR":STOP
100 DATA120,173,20,3,174,196,3,1
41,196
110 DATA3,142,20,3,173,21,3,174,
197,3
120 DATA141,197,3,142,21,3,169,2
55,160
130 DATA0,145,251,136,208,251,88
,96,8
140 DATA72,152,72,138,72,169,255
,141,1
150 DATA221,162,7,169,255,24,106
,72
160 DATA188,0,48,49,251,145,251,
104
170 DATA202,16,243,160,48,136,20
8,253
180 DATA169,255,160,0,49,251,141
,1,221
190 DATA200,208,248,162,7,169,25
5,188
200 DATA0,48,145,251,202,16,248,
104
210 DATA170,104,168,104,40,76,49
,234
220 DATA13072:REM*CHECKSUM*
```

BASIC Calling Program

```
10 REM **** MULTIPLE SERVO ****
20 :
30 DN=8:REM IF CASSETTE THEN DN=
1
40 IF A=0 THEN A=1:LOAD"MULTISER
V.HEX*",DN,1
50 POKE 778,88: POKE 779,3:REM P
OINTER TO EVENT HANDLER
60 POKE 251,0: POKE 252,49:REM S
ET UP ZERO PAGE PTR
70 DDR= 56579:POKE DDR,255:REM A
LL OUTPUT
80 MC=820: SYS MC
90 :
100 GET K$:IF K$="" THEN 100:REM
 AWAIT KEY
110 REM ** ALTER MOTOR POSITION
**
115 AK=ASC(K$)
120 IF AK>48 AND AK<58 THEN POKE
12288+SERVO,VAL(K$)*20
130 IF AK>32 AND AK<40 THEN SERV
O=ASC(K$)-35
140 IF K$<>"E" THEN 100:REM 'E'
TO EXIT
```

Single Servo Motor Control

```
 755 REM ****************************
 756 REM   Assemble the machine code
 757 REM ****************************
10000 DEF PROCinitial
10010 DIM space% 200
10020 FOR C=0 TO 2 STEP 2
10030   portb=&FE60
10040   P%=space%
10050   angle=P%
10060   P%=P%+1
10070   [OPT C
10080   .eventhandler
10090   \save registers first
10100   PHP:PHA:TYA:PHA:TXA:PHA
10110   LDA &&04
10120   LDX &ytimer
10130   LDY &ytimer
10140   JSR &FFF1 \reset timer
10150   LDA &&FF
10160   STA portb
10170   \wait approx. 1msec
10180   LDY &&FF
10190   .LOOP DEY
10200         BNE LOOP
10210   \and countout pulse
10220   LDY angle
10230   .LOOP1 DEY
10240         BNE LOOP1
10250   \stop all output pulses
10260   LDA &&0
10270   STA portb
10280   PLA:TAX:PLA:TAY:PLA:PLP
10290   RTS
10300   ]
10310   NEXT C
10320 REM point to eventhandler
10330 !&220=eventhandler OR (!&220 AND &FFFF0000)
10340 ENDPROC
```

BBC Micro Listings

Common Initial Routine

```
 10 MODE 0
 15 REM ****************************
 16 REM      Set up the timer etc
 17 REM ****************************
 20 osbyte=&FFF4
 30 A%=&97 :X%=&62 :Y%=&FF
 40 CALL osbyte:REM set up port B for output
 50 CLS
 60 DIM p%(8)
 70 DIM timer% 12 ,read% 12
 80 xtimer=timer% MOD 256
 90 ytimer=timer% DIV 256
100 xread=read% MOD 256
110 yread=read% DIV 256
120 PROCinitial
130 FOR I%=angle TO angle+8:angle?I%=128:NEXT
140 t=.02  :REM sec between pulses
150 time%=&FFFFFFFF -(t*100) +1
160 timer1?4=&FF    :REM load highest byte
170 !timer%=time%   :REM set up timer, enable events
180 *FX14,5
190 A%=4 :X%=xtimer :Y%=ytimer :CALL &FFF1
195 REM ****************************
196 REM     The BASIC controller
197 REM ****************************
200 CLS
210 PRINT"PRESS  SHIFT + NUMBER to select motor"
220 PRINT"PRESS  NUMBER to select angle"
225 motor=1
230 REPEAT
240   A=GET
250   IF A%>2F AND A<&3A THEN a=(A-&30)*10#(255/90):angle?
      (motor-1)=a:PRINT'&&B:10,motor);"motor  motor " angle "a#(90/255)
260   IF A%>20 AND A<&2A THEN motor=A-&20
270   UNTIL 0
280 END
750
```

Multiple Servo Motor Control

```
755 REM ****************************
756 REM Assemble the machine code
757 REM ****************************
760 DEF PROCinitial
770 DIM space% 600
780 FOR C=0 TO 3 STEP 3
790   zeropage=&70   :REM free for users
800   portb=&FE60 :osword=&FFF1
810   P%=space%
820   angle=P% :P%=P%+8  :REM potentially 8 motors
830   table=P% :P%=P%+256  :REM 256 possible pulse lengths
835   FORI%=table TO table+&100:?I%=&FF:NEXT
840   lowtable%=table MOD 256
850   hightable%=table DIV 256
860   ?zeropage=lowtable% :zeropage?1=hightable%
870   [OPT C
880   .eventhandler
890   PHP:PHA:TYA:PHA:TXA:PHA
900   LDA &&04
910   LDX &xtimer
920   LDY &ytimer
930   JSR osword
940   \Start pulse, for some motors it may be possible to start
1000  \before filling table and so reduce the wait loop below
1010  LDA &&FF :STA portb
1020  \fill table with exceptions
1030  LDX &&7 :LDA &&FF :CLC \set up bit pattern
1040  .exceptions
1050        ROR A :PHA  \bit pattern
1060        LDY angle,X \get offset corresponding to angle of motor X
1070        AND (zeropage),Y \keep existing bit pattern
1080        STA (zeropage),Y \but modified for motor X
1090        PLA :DEX
1100        BPL exceptions
1110  \table is now loaded, fill in some time
1120  LDY &&60
1130  .wait DEY
1140        BNE wait
1150  LDA &&FF  \all pulses on
1160  LDY &&0
1170  .loop AND (zeropage),Y  \but mask off with each table
1180        STA portb  \element in turn
1190        INY
1200        BNE loop
1201  LDX &&7 :LDA &&FF
1203  .clear
1205        LDY angle,X  \clear all the exceptions again
1207        STA (zeropage),Y
1208  DEX
1209  BPL clear
1210   \all pulses should now be finished
1220  PLA:TAX:PLA:TAY:PLA:PLP
1230  RTS
1240  ]
1250  NEXT C
1260 !&220=eventhandler OR (!&220 AND &FFFF0000)
1270 ENDPROC
```

PLAYSCHOOL

This is the second of two articles in which we sample a variety of educational programs for very young people. Here, we consider the merits of two packages from Spinnaker, and look at a series of programs written by the educational publishers Macmillan in collaboration with Sinclair Research.

Story Machine (Spinnaker, £9.95) is designed for children between the ages of five and nine. It is intended to teach the rules of syntax, improve spelling and stimulate expressive writing.

The program contains a dictionary of 52 words, of which five are proper nouns — the names of a boy, a girl, a cat, a dog and a creature called a 'bumpus', all of which must be specified by the user at the beginning of the program.

From this dictionary, the child builds up simple sentences that are then acted out by the characters

Story Machine

on the screen. For example, the sentence 'Maurice kisses flowers' will display a character showing a boy (who has been named Maurice by the user), a character depicting some flowers, and, when they get close together, flashing hearts appear to indicate the kissing.

If a word is misspelt or wrongly used, then the computer will refuse to accept it, displaying a message explaining what is wrong and asking for the word to be altered or changed. At the end of each sentence — that is, after each full stop — the program will attempt to enact it. When a whole story has been made up, there is an option to replay it in its entirety.

If you get bored writing your own stories, Story Machine has two other options: you can share the story writing with the program (taking turns to add words); or you can let the computer do all the work, and write the story for you. This is possible because the dictionary is divided up into parts of speech — nouns, verbs, prepositions, etc. — and it has a strict grammatical algorithm to generate its own sentences.

Most of the words in the dictionary, as well as the sentence structures used by the program, are simple enough. The verbs in particular describe easily-depicted actions, such as 'hop(s)' and 'go(es)'. It is interesting that an imaginary verb 'zot' — which appears to mean hit or strike — has been included. This seems to be at variance with Spinnaker's stated rule that it is wrong to hurt other people or animals, and therefore any actions that involve physical injury to others will not be allowed by the program.

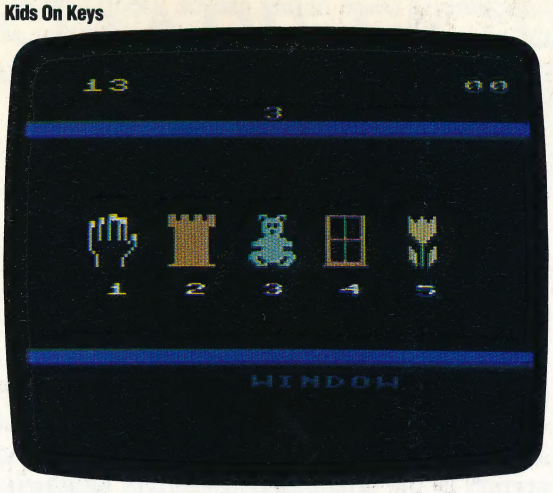
The program has two fundamental grammatical rules: a sentence must contain a subject, a verb and an object, in the correct order; and singular and plural forms must be consistent. Beyond these rules, the enforcement of syntax is less satisfactory, because of the enormous variations of usage in the English language. For example, the computer will accept the sentence 'Houses eat rocks' but will not accept the sentence 'Girls eat rocks' without the definite article before the word 'girls'.

The program even appears to break its own rules. After the sentence 'Bumpuses eat rocks', the program generated the sentence 'They walk to its fence'. Normally, when encountering the word 'its', the computer would generate the message 'Whose?' and ask for the word to be changed. The program has also ignored its rule of not mixing singular and plural forms. These may be considered minor errors, but a program that claims to be educational should at least be consistent and *must* always be correct.

Kids On Keys (£9.95) is another Spinnaker product, designed for children aged from three to nine. The aim of the program is to teach the user how to identify words, letters and numbers. There are three different games in the package, each of which has various levels of difficulty. In the first game, a letter scrolls down the screen, and the child has to press the correct key before the letter reaches the bottom. After 15 letters, a balloon floats down containing a word. This must also be typed in correctly before the balloon reaches the bottom of the display. The game is, at least, an enjoyable way for a child to become familiar with the layout of the keyboard.

In the second game, a series of pictures (expanded sprites) scrolls down the screen, and the child has to type in the name of the object before it reaches the bottom. After this, there is a

Kids On Keys



bonus round in which the same objects descend, but these have two quarters of the image missing, making recognition much more difficult. At this point, the game tends to veer towards being a test of reflex action as the player frantically searches for the correct letters to type. A major drawback with the program is the absence of a delete facility, which makes a false keypress extremely frustrating.

Perhaps the most serious problem with the game is that some of the sprites are badly drawn and difficult to identify. For instance, if a child decides that a particular drawing is a man, he or she may type in the correct spelling of 'man', only to find that the sprite continues its descent because the program regards it as a 'bear' or a 'boy'. The child may then conclude that its spelling was at fault, since the program makes no attempt to explain the reason for a failure (as it has no facility to analyse this). Furthermore, once the sprite has reached the bottom of the screen, the program gives no indication of what the correct answer was. A good educational program would not only give the user another attempt at the problem, but would also display the correct answer after several false responses had been attained.

In the third game, five pictures and a word are displayed on the screen, and the child has to match the correct picture with the word. This game is less speed-oriented than the others, although it also suffers from poor graphics.

LEARN TO READ

The UK educational publishing house Macmillan has developed a series of programs for the Spectrum under the collective title Learn To Read. These five programs — produced in conjunction with Sinclair Research — are derived from Macmillan's highly successful Gay Way reading scheme, a child-centred course that is widely used in primary schools. Because the Gay Way course is oriented towards the individual, it has the advantage that its users learn to read at their own speed. A series of computer programs is a natural extension of this, since using a computer is clearly an individual-oriented activity.

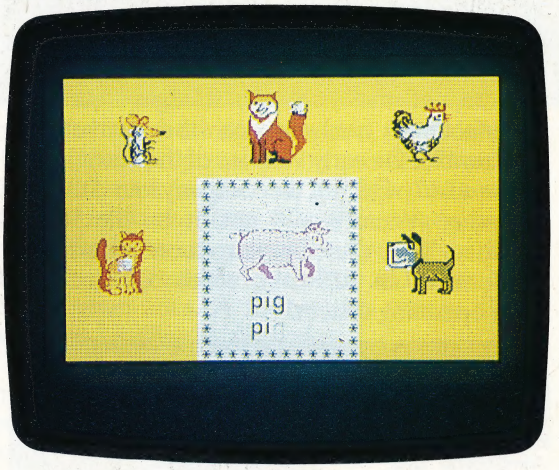
The five packages make up an entire preliminary reading course. The first program introduces six animals (Deb the rat, Sam the fox, etc.) and these creatures are featured throughout the course to develop a sense of continuity and familiarity.

The programs assume that the child has no previous knowledge of computers or reading. Selecting one of the options within a program is straightforward. For instance, the first program displays five options, and a flashing rectangle appears around the name of each of these options in turn. The user waits until the rectangular 'cursor' surrounds the desired option and then presses any key on the keyboard. No other instruction is needed for using this menu.

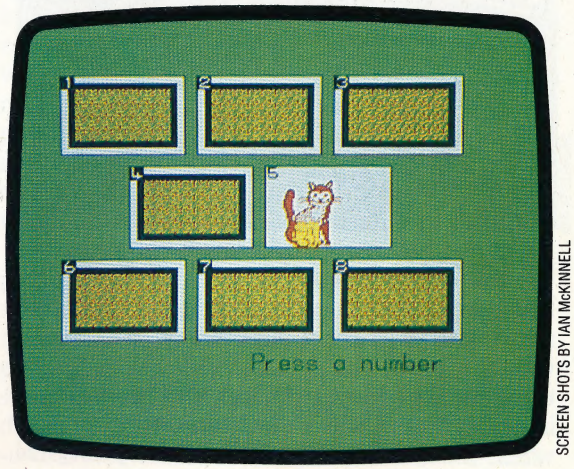
Once an option has been chosen, the computer demonstrates what is required, and then prompts the child to make a response. If the child makes a false response, he or she will be given several more opportunities before the computer displays the correct answer.

The six animals used in the programs are drawn in high resolution graphics, and these are far more attractive than the whizz-bang sprites used in some of the other educational games we looked at. The entertainment value of the Learn To Read series may not be nearly as great as many other packages, but their educational value would appear to be far greater — and in the long term perhaps far more rewarding to the child.

Learn To Read



Learn To Read



SCREEN SHOTS BY IAN McKINNELL

K

KEYPUNCH

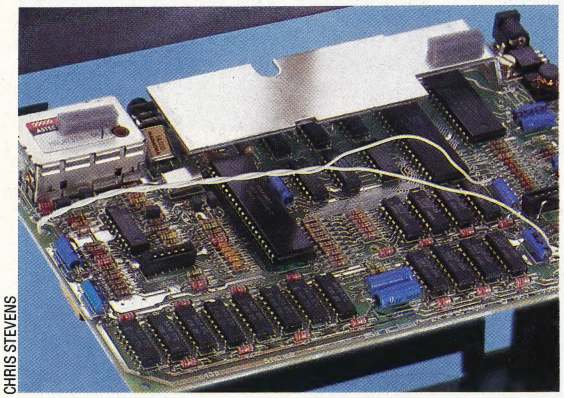
Keypunch is a data entry system that relies on holes punched in specified locations on a rigid paper card. Using the Hollerith code (see page 769), individual letters and numbers can be represented by a series of strategically punched holes on a card divided into 12 rows of 80 columns. The keypunch system was the primary means of entering large amounts of data into a computer system, and is still widely used in data processing applications.

KEYWORD

To resemble human language, and thereby make computer programming more 'user-friendly', higher-level programming languages contain a vocabulary of words whose precise meaning is a part of the structure of the language. When the language processor (compiler or interpreter) encounters a *keyword*, it accesses a predefined procedure or routine. In BASIC, keywords form the command structure, so that words like PRINT, LET, GOTO, and IF...THEN have specific functions. Because of their specialised meanings, keywords cannot be found in, or used as, variable names.

KLUDGE

A *kludge* is a colourful term for a solution to a problem — usually a hardware problem — which arises after a system has been designed and largely executed. Rather than redesign the entire system, a kludge is hastily constructed from available parts and added to the system as cleanly and quietly as possible. Forgivable, and often creatively-inspired, in the hobbyist, kludges found in manufactured hardware on the market are inexcusable. On early versions of the Sinclair QL, reviewed on page 501, the company found that the operating system would not fit on the ROMs provided. They therefore introduced a kludge, in the form of an EPROM, which contained the extra operating system.



CHRIS STEVENS

Flying Kludge
The obtrusive wires attached to the Spectrum PCB connect an additional reset switch into the logic, thus turning the Spectrum into the Spectrum+ — a fine example of kludging.
The extremely expensive alternative would have been to redesign the PCB entirely, incorporating the new wires into the copper track patterns on the board

LABEL

Names, or *labels*, serve to identify an object and set it apart from other objects. Labels occupy a variety of positions in computer operations, so the term has several distinct usages, the most important of which is in Assembly language, where labels are names attached to individual instructions. These names can be used to mean the address of those instructions; for example:

Label Field	Instruction Field	Operand Field
LOOP	LDA	TEST
EXIT	BNE	LOOP

Here, LOOP, TEST and EXIT are all labels; they are Assembly language's equivalent of variables. When the program is passed to the assembler for conversion to machine code, the labels are replaced by the addresses they represent.

One form of label is a statement, such as a REMark statement in BASIC or a coded phrase within the body of a computer program, which identifies the specific task to be carried out by the portion of the program to which it is attached. Labelling in this sense is a key element of structured programming in that it helps the programmer to keep track of what the program is actually doing at every stage.

Structured programming also makes use of procedures. These are essentially subroutines that are labelled and which can be referred to and acted on by their labels. For example, the commands required to create a graphics backdrop in a game program might form a procedure called SCREEN. When the program encounters the word SCREEN, it executes all the commands pointed to by that label.

A label is also a small file on a larger magnetic tape or disk data file that identifies the nature of the data held in the file. A volume label identifies the contents of the entire tape or disk file. Header labels directly precede and characterise individual data files.

Finally, some sophisticated spreadsheet programs have the facility of labelling related regions of cells, then performing calculations on the entire region simply by incorporating the label into a formula. For example, actual sales figures of a product are recorded for a one-year period. To predict what sales will be for the same period one year in the future, a formula calculates a given percentage of growth, or decline, and is applied to the actual sales numbers. Without labelling, making this happen would require including the row and column number for each actual figure, like this:

```
A4 * 110% = B4
```

and the formula must then be copied to the relevant portions of the spreadsheet, requiring several steps. With a labelling facility, however, the same formula looks like this:

```
Actual Sales * 110% = Projected Sales
```

with labels identifying the significant regions. This formula now applies automatically to all of the cells in the area labelled 'Actual Sales', with the results automatically being assigned to the region labelled 'Projected Sales'. This speeds up the creation and operation of the spreadsheet and makes the results much clearer to read.

SHARP SIGHTED

Two Japanese companies, Casio and Sharp, produce computers small enough to fit into a pocket. Despite the size of these machines, they are true computers, although their specification is limited. We have already taken a close look at Casio's range of pocket computers (see page 541); now we scrutinise Sharp's machines.

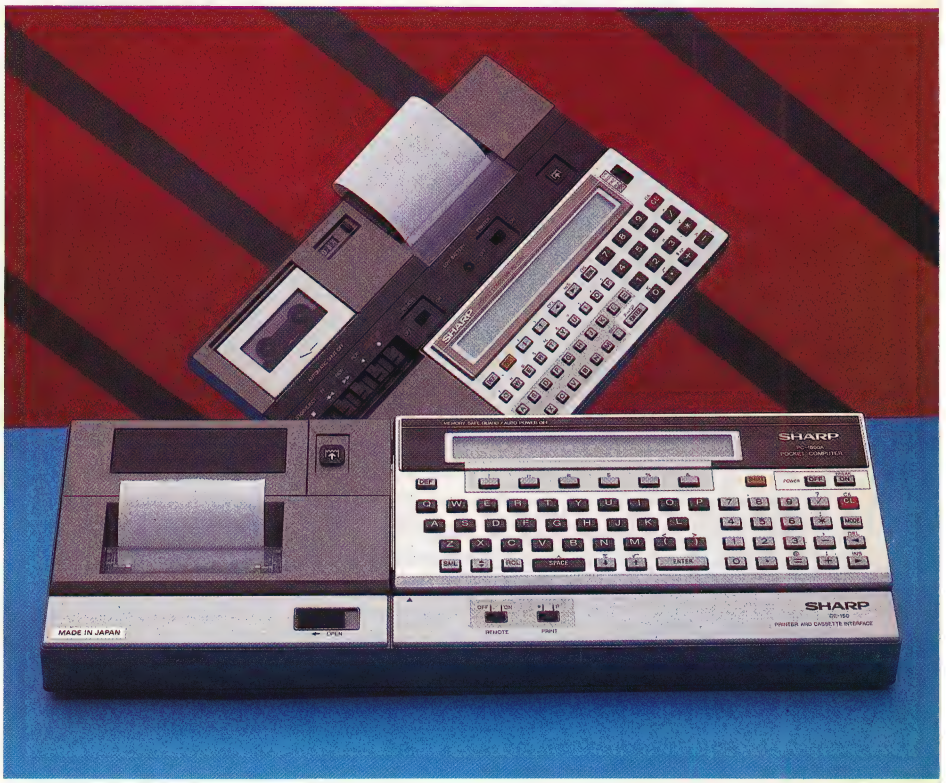
Sharp's marketing strategy is in marked contrast to Casio's. To begin with, it produces two quite different pocket computers, while Casio markets three relatively similar machines. Sharp's two machines are designed not to compete with each other: the PC-1251 is the smallest pocket computer currently available, and costs £80; the PC-1500A is considerably larger and more powerful, and is the most expensive pocket computer (£170) offered by either company.

The Sharp PC-1251 weighs a mere 115 grams (four oz) and measures 135 by 70 by 10 mm ($5\frac{1}{2}$ by $2\frac{3}{4}$ by $\frac{1}{2}$ in). The keyboard is well under half the size of a standard keyboard and the keys are only four mm ($\frac{1}{8}$ in) wide. They are big enough to ensure that a finger pressing one key does not press the four around it, provided care is taken. At the side of the alphabetic keyboard is a number pad with larger keys, which allows the computer to be used as an ordinary pocket calculator.

The PC-1251 has a 24-character wide liquid crystal display. This can be adjusted for different viewing angles and lighting conditions by a small wheel on the side of the computer. To the right of the display is a Mode Selector switch. There are three operational modes: one allows the functions of certain keys to be defined (RSV); a second facilitates programming (PRO); and the last allows a BASIC program to be run or the machine to be used as a calculator (RUN). This switch is also used to turn the machine off.

Although not quite up to the standard of ordinary home micros, the version of BASIC used is good for a machine so small. It has commands that some of the Casio pocket computers lack, such as ASC and CHR$, but like them it does not have the ELSE option for the IF. . . THEN instruction. Like the smaller Casio computers, the PC-1251's BASIC uses the same single letter names for strings and variables. This means that if the variable A was used to hold a number, then the string variable A$ could not be used. In the same way, some arrays could overwrite the same areas of memory.

Only nine error messages are produced by this version of BASIC and these are all single letters, which is extremely unhelpful for isolating bugs in programs. As an additional option, the command PASS allows programs to be protected with a password. Line numbers in the PC-1251's BASIC are limited to the range 1 to 999. When in the mode for entering programs, two cursor keys allow the programmer to scroll up and down over the program lines. A further two cursors allow sideways scrolling in all modes.



Pocket Power
With CMOS RAM and 8-bit CPU, QWERTY keyboard, BASIC and a range of optional peripherals, these Sharp 'calculators' can reasonably claim to be small-scale and truly portable microcomputers

As there is very little software available for the machine, most users will have to write their own programs. The machine's manual helps here in two ways. First of all, it gives a good and easily understood guide to the BASIC, although it does not include a tutorial for the beginner. Secondly, it contains listings for nine short programs, not all of which are mathematical applications (finding roots, standard deviation, etc.) — a 'typing practice' program and a 'soft landing' game are among others included. Furthermore, Sharp offers three tapes with a selection of programs on them for £14 each.

Whereas the Casio computers can have up to 10 programs in memory at once, the PC-1251 is restricted to one at a time. However, it is possible to use one program made up from several sub-programs, each separated by END statements. A sub-program could then be run by using GOTO with the appropriate line number. It is useful to have several programs in memory at once, since there is

no way of loading programs, other than keying them in each time they are needed. Furthermore, the computer has only four Kbytes of memory, so there is little room for more than a few modest BASIC programs. This means that for almost any serious use the optional printer and cassette add-on is essential.

SHARP PC-1500A

Sharp's other portable computer, the PC-1500A, is a development of an earlier model (the PC-1500) and is designed with the more serious user in mind. It costs £170, although with all of the main accessories included the price leaps to £735. The PC-1500A measures 195 by 85 by 25 mm (8 by 3½ by 1 in) and weighs 375 gm (13 oz), which is over three times heavier than the PC-1251.

The PC-1500A has a better keyboard and a slightly larger display than its sibling. The width of the LCD is minimally expanded — 26 characters instead of 24 — and the PC-1251's handy screen adjustment facility has disappeared from the larger machine.

The version of BASIC, however, is a great improvement over the other. Variables can have two-letter names, and the same name can be used for string and numeric variables without causing confusion. The BASIC also supports some useful sound and graphics facilities. Patterns with a resolution of seven rows by 156 columns can be drawn on the LCD, using the GPRINT command to define each seven-dot column. The machine's BEEP command allows control over the pitch and duration of notes, which are reasonably loud.

The computer has a built-in calendar and clock, which can be accessed using the variable TIME. The computer keeps the time even when it is turned off, so once set it keeps ticking over. This facility would be a useful addition to any home micro. The PC-1500A has several other commands in its BASIC that give it an advantage over many home micros. These include an ON ERROR GOTO command and trace facilities — TRON and TROFF. It has a generous 39 error messages for the standard computer; another 16 are available for commands used only with add-on units. However, like the PC-1251, these error messages are given as numbers, and could be more helpful.

The top row of alphabetic keys have BASIC keywords programmed into them, and a plastic template can be fitted over them to identify the 10 commands. Why Sharp chose not to print these commands directly on the casing above the keys is a mystery — the template is easily lost. The six keys above the main keyboard can have up to 18 functions programmed into them.

PC-1500A users will have to write most of their own software: only a few companies produce programs for the machine and Sharp itself sells only one tape of assorted programs for £15. A book supplied with the machine lists 53 programs, with a variety of applications in five main areas — mathematics, statistics, electrical, office work and games. These programs were originally written for the PC-1500, but they all work perfectly well on the PC-1500A because the only difference between the two models is the amount of RAM memory available. The earlier model had 3½ Kbytes of memory, while the PC-1500A has 8½ Kbytes.

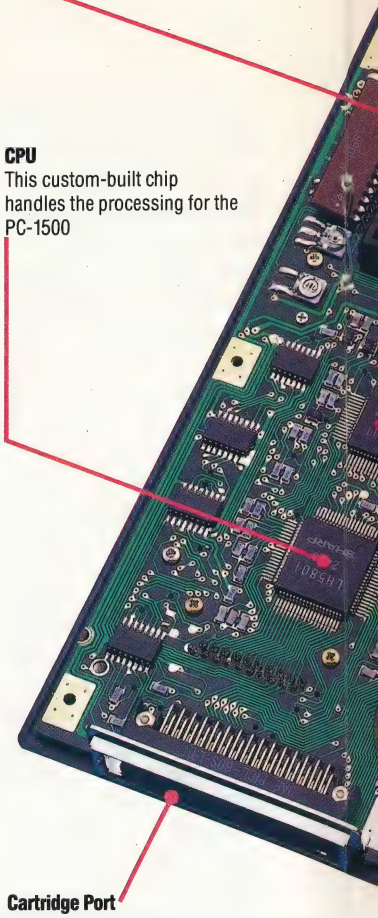
Using a cartridge slot on the underside of the machine, it is possible to increase the amount of memory available. Four memory cartridges are offered, and all of them are rather expensive. The standard four Kbyte and eight Kbyte memory packs (priced at £50 and £80 respectively) hold their contents only while they are in the machine. Two other cartridges have batteries in them so that their contents can be saved even when they are removed from the computer. These have capacities of eight Kbytes (£90) and 16 Kbytes (£110).

Priced at £150, the printer/plotter and cassette interface unit is much better value for money. The cassette interface allows an ordinary cassette recorder to save and load programs — and Sharp offers a compatible cassette recorder of its own for £40. The printer/plotter part of the unit uses four ball pens to draw good quality letters and graphics in four colours. It is almost identical to a number of other printer/plotters on the home computer market (see page 289), but it uses paper only 57 mm (2¼in) wide, which is a great limitation.

The BASIC includes a full set of commands to use the printer/plotter. These are: CSIZE to produce different-sized letters, ROTATE to print characters on their sides or upside down, COLOR to select the pen colour, LF to move the paper up and down, LPRINT to print text, LCURSOR and GLCURSOR to move the pen in text and graphics mode, SORGN to set the origin, and LINE and RLINE to draw a line between two points using absolute and relative co-ordinates respectively. The printer/plotter and cassette unit is supplied in its own case, along with accessories such as a mains transformer (with a lead for an ordinary mains plug) to power its rechargeable batteries.

Two other major add-ons are produced for the PC-1500A. One is a Centronics and RS232 interface, which allows the machine to communicate with full-size printers and computers. The other peripheral is called a 'software board'. This device is a large touch-sensitive pad, which has 140 definable keys that can be programmed to perform commonly-used tasks (such as automatically calculating totals in spreadsheet applications). As the software board costs £80, and requires the interface (£150) for its operation, the price of this expansion facility is rather prohibitive.

Although the PC-1500A can be expanded into a powerful system with many impressive facilities, many similarly-priced micro systems are more versatile and — most important of all — supported by a wider range of software. By attempting to top the lightweight computer league, the PC-1500A is in danger of being heavily outclassed by a lot of more powerful middleweights.



Power Supply Jack
As an alternative to using batteries, the computer can be run from the mains using a suitable transformer

BASIC ROM
This chip holds the Microsoft BASIC used in the computer

Extra Sharp
The Sharp PC-1251 fits into the CE-125 printer/microcassette unit (£100). This contains a thermal printer (24 characters per line) and microcassette recorder. The unit measures 205 by 149 by 23 mm.

The Sharp PC-1500 is the centre of an extended family of Sharp equipment, and interfaceable through the CE-158 RS232C/ parallel interface (£150) to almost any micro or mainframe system. The CE-150 colour printer/cassette interface (£150) is a four-colour X-Y plotter with nine type sizes and graphics option. The cassette interface allows connection to two cassette recorders.

The CE-151, CE-155, CE-159 and CE-161 memory modules (£50 to £110) are a range of 2 to 16 Kbyte CMOS RAM packs, some containing programmable ROM. The CE-153 Software Board (£80) is a 140-key touchpad for formatted input.

SHARP PC-1500A

PRICE
£170

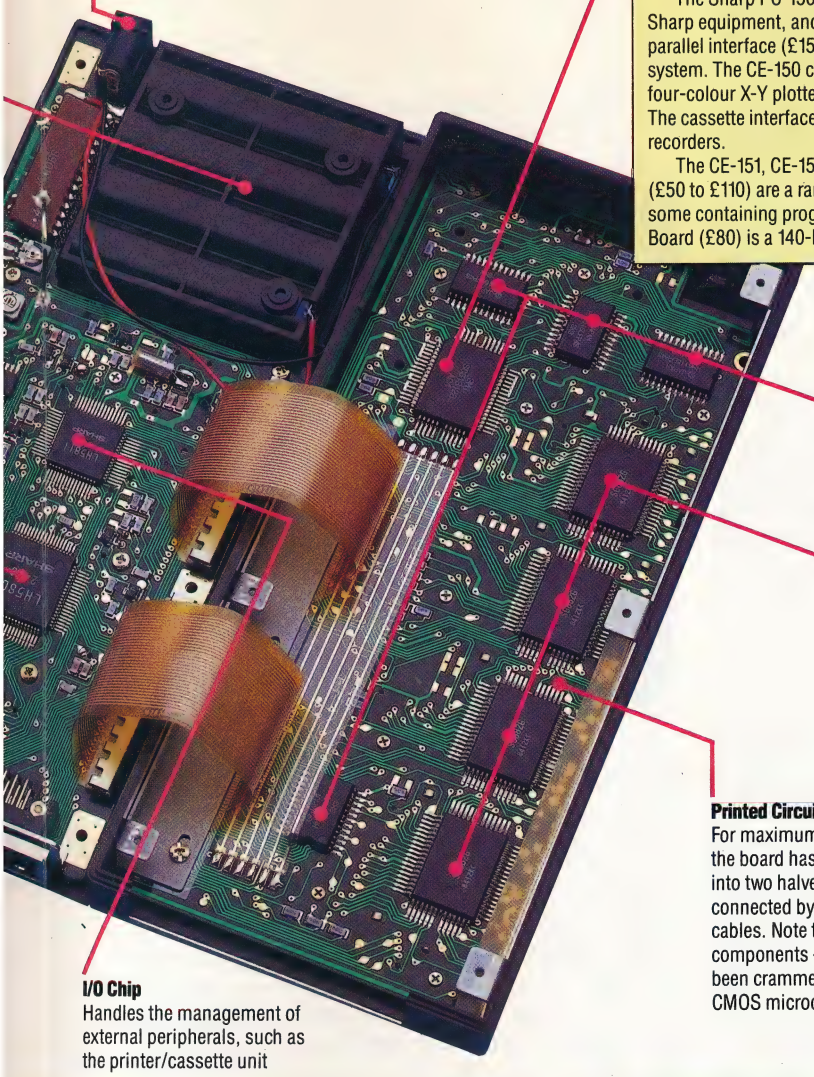
DIMENSIONS
195×85×25mm

WEIGHT
375 gm

MEMORY
8.5K RAM

SCREEN
1×26 character LCD

REMARKS
Built-in calendar and clock; good BASIC; expandable memory; large range of peripherals



RAM Chips
These chips provide the 8·5 Kbytes of memory, of which 6.6 Kbytes are available to the user

Display Chips
These four chips handle the display for the LCD screen

Printed Circuit Board
For maximum compactness, the board has been divided into two halves, which are connected by a pair of ribbon cables. Note the lack of components — these have been crammed onto the CMOS microchips

I/O Chip
Handles the management of external peripherals, such as the printer/cassette unit

SHARP PC-1251

PRICE
£80

DIMENSIONS
135×70×10mm

WEIGHT
115 gm

MEMORY
4K RAM

SCREEN
1×24 character LCD

REMARKS
Three operations modes; 18 user-definable keys; cassette/printer available

Sharp PC-1251

This calculator is seen by Sharp as their workaday business system — probably more for the engineer and scientist than the administrator — the tiny letter keys on its QWERTY keyboard make text entry difficult and tiring

Sharp PC-1500A

This is an impressively flexible pocket computer in calculator size; its power and optional equipment are its selling points, and it could make a real contribution to coping with the office workload. However, it is likely to be seen as merely an executive toy and impressive calculator

CHRIS STEVENS

DRAWING PICTURES

In the last instalment of our adventure game programming project, we looked at the design of two graphic screens for the BBC Micro. These screens depicted two locations of importance in our Digitaya game — the ALU and the joystick port. We now look at the design of the same screens for the Sinclair Spectrum.

The design of the ALU screen involves the scrolling of the letters A, L and U down to the centre of the screen using high resolution graphics. On the BBC Micro, this scrolling was performed by drawing the letter from a specified start point using relative drawing commands, then rubbing it out, moving the start point and repeating the whole procedure (see page 904). The same idea can be used in the Spectrum version.

The Spectrum's DRAW command allows relative drawing only — that is, starting from the last point specified — but this is ideally suited to this particular scrolling application. By PLOTting an initial start point and then carrying out a series of DRAW commands to create the shape of the letter, we can easily move the entire letter design around the screen by simply changing the co-ordinates of the point initially PLOTted. Rubbing out can be accomplished by drawing the same shape in the same position, but with all the colours inverted. This effect is turned on by using INVERSE 1, and turned off again with INVERSE 0. Thus, for each position that the letter takes up, we shall draw it twice: once with INVERSE 0, to make the shape appear, and again with INVERSE 1 to rub it out.

If we take the example of the letter A, which scrolls on from the left, we can place all these instructions within a FOR . . . NEXT loop. This loop increases the value of the x co-ordinate of the initially-plotted point for the shape. Nested inside this loop is a second FOR . . . NEXT structure that simply carries out the drawing commands twice. The last value of x is 55, which denotes the final resting position of the letter on the screen. Obviously, we do not want to rub out the final version of the letter, so a test is inserted to ensure that the letter will be erased (by switching to INVERSE 1) only if the x co-ordinate is less than 55. The principles discussed here are also applied to the other two letters to make L scroll up the screen and U scroll on from the right.

ALU ROUGH DESIGN

When designing a graphic screen it is important to rough out a design on paper and make an initial estimate of the co-ordinate values that each shape on the screen will have. In addition, any letters that have to be PRINTed to the screen should also be positioned, in terms of rows and columns. The screen shot shows such a design, with the screen dimensions in graphics and character units.

IAN McKINNELL

The words AND, OR and NOT are positioned on the screen using the PRINT AT r,c command: r being the number of rows from the top of the screen, and c indicating the number of columns from the left hand margin. The buttons are drawn using CIRCLE x,y,r where the co-ordinates of the centre and the length of the radius are specified.

On completion of the drawing routines, the program waits for a keypress before resetting the INK and PAPER to the original colours and clearing the screen. It then RETURNs to the main ALU routine. The keypress is tested by INKEY$; if no key is pressed then the test is simply repeated.

To call this subroutine, the following line should be inserted into the Digitaya program:

```
4565 GOSUB 7000:REM ALU PICTURE S/R
```

THE JOYSTICK PORT

The joystick port screen is designed to shoot laser beams from the centre of a joystick connector socket. The pins for the socket are full stop characters PRINTed to the screen and the D-type surround is drawn using high resolution graphics. To give the picture a sense of depth, a series of tapering lines is drawn in the foreground. The start point for each line is on the horizon line, and is selected by a PLOT command. The end of each line is at the bottom of the screen. The lines are spaced at one-unit intervals on the horizon, and widen to seven units at the bottom of the screen.

The fact that Spectrum BASIC's DRAW command is relative makes the routine slightly more complex than if we were able to specify an absolute end

ERRATA

1) In the program at the top of the right-hand column on page 636, the last condition in lines 260 and 370 should read:

```
OR fire=1
```

2) In the program on page 700, the last statement in line 30210 should read:

```
:GOTO 30050
```

3) In Basic Flavours on page 848, replace the first Flavour by:

Replace SN$ by SS, IVS(,) by VS(,), ICS() by IS() and NN$ by R$

point. If the x co-ordinate of the start point of the left-most line is 111, then we have to make a calculation based on this to find the relative offset to the end point. The FOR . . . NEXT loop in lines 8030-8060 shows this calculation, which is based on the step-length at the bottom of the screen and the start point on the horizon.

As before, it is useful to rough out the dimensions and co-ordinates of the design on paper before starting to write the code. The screen shot shows such a design:

The laser beams are drawn from the joystick port by moving to a point at the centre of the port and then DRAWing a line to a random point on the horizon, using a randomly-selected INK colour. By repeating the procedure with INVERSE 1, we can rub out the line, making the beam appear for only a short interval (creating a lightning effect). However, when the line is rubbed out a problem occurs. Because the beam is drawn from a point at the centre of the port, it crosses the previously drawn graphics that depict the port itself. When the line is rubbed out, gaps appear in the joystick port graphics, and so it is necessary, when rubbing out the line, to redraw them.

Even though the end of any line drawn from the joystick port stops just short of the horizon line, the horizon is also affected. Because of the way the Spectrum controls colour, the portion of the horizon nearest the point where the beam ends takes on the same colour as that used to draw the line. This is because the Spectrum can support only one INK and one PAPER colour within any one character cell; any graphics already present within the cell take on the foreground colour of the new INK colour used in the cell. Therefore, in addition to redrawing the joystick port, the horizon line must also be redrawn after a beam line is erased. The routine continues to fire laser beams until a keypress is made, at which point program control is RETURNed to the main joystick port routine, after having reset the normal INK and PAPER colours.

The following line should be inserted in the main program to call this subroutine:

```
3845 GOSUB 8000:REM JOYSTICK PORT PICTURE
```

Implementing these two graphic screens on the Commodore 64 will be the subject of the next instalment in the project.

ALU Screen

```
7000 REM **** alu picture s/r ****
7010 INK 6: PAPER 0: CLS
7015:
7017 REM **** letter A ****
7020 FOR x=0 TO 55 STEP 5
7030 INVERSE 0
7040 FOR i=1 TO 2
7050 PLOT x,100
7060 DRAW 0,30
7070 DRAW 15,20
7080 DRAW 15,-20
7090 DRAW 0,-30
7095 DRAW 0,20
7096 DRAW -30,0
7110 IF x<55 THEN  INVERSE 1
7115 NEXT i
7120 NEXT x
7130:
7140 REM **** letter L ****
7150 FOR y=100 TO 150 STEP 5
7152 INVERSE 0
7155 FOR i=1 TO 2
7160 PLOT 113,y
7170 DRAW 0,-50
7180 DRAW 30,0
7190 IF y<150 THEN  INVERSE 1
7200 NEXT i
7210 NEXT y
7220:
7230 REM **** letter U ****
7240 FOR x=225 TO 170 STEP -5
7250 INVERSE 0
7260 FOR i=1 TO 2
7270 PLOT x,150
7280 DRAW 0,-50
7290 DRAW 30,0
7300 DRAW 0,50
7310 IF x>170 THEN  INVERSE 1
7320 NEXT i
7330 NEXT x
7340:
7350 REM **** buttons ****
7360 PRINT AT 10,7;"AND"
7370 PRINT AT 10,15;"OR"
7380 PRINT AT 10,22;"NOT"
7390 INK 3: CIRCLE 70,80,5
7400 INK 4: CIRCLE 128,80,5
7410 INK 5: CIRCLE 185,80,5
7420:
7430 REM **** q mark ****
7435 INK 6
7440 PLOT 113,45
7450 DRAW 0,15
7460 DRAW 30,0
7470 DRAW 0,-20
7480 DRAW -15,0
7490 DRAW 0,-7
7500 FOR r=6 TO 0 STEP -2
7510 CIRCLE 128,23,r
7520 NEXT r
7530:
7540 IF INKEY$="" THEN  GO TO 7540
7550 INK 0: PAPER 7: CLS
7560 RETURN
```

Joystick Port Screen

```
8000 REM **** jstick port pic s/r ****
8010 INK 6: PAPER 0: CLS
8020 REM **** foreground ****
8030 FOR n=1 TO 31
8040 PLOT 112+n,50
8050 DRAW 7*n-112,-50
8060 NEXT n
8070:
8080 REM **** horizon ****
8085 INK 6: INVERSE 0
8090 PLOT 0,50
8100 DRAW 255,0
8110:
8120 REM **** port ****
8130 PRINT AT 1,18;"JOYSTICK PORT"
8140 PRINT AT 3,20;". . . . ."
8150 PRINT AT 5,21;". . . ."
8160 PLOT 158,152
8170 DRAW 75,0
8180 DRAW 1,-1
8190 DRAW 1,-1
8200 DRAW 0,-1
8210 DRAW -1,-1
8220 DRAW -10,-25
8230 DRAW -2,-2
8240 DRAW -52,0
8250 DRAW -2,2
8260 DRAW -10,25
8270 DRAW -1,1
8280 DRAW -1,1
8290 DRAW 0,1
8300 DRAW 1,1
8310:
8320 REM **** shoot ****
8340 INK RND*7
8350 LET x=RND*255-194
8360 LET y=-86
8365 INVERSE 0
8367 FOR i=1 TO 2
8370 PLOT 194,136
8380 DRAW x,y
8385 INVERSE 1
8387 NEXT i
8390:
8400 REM **** test for key ****
8410 IF INKEY$="" THEN  GO TO 8080
8415 INVERSE 0
8420 INK 0: PAPER 7: CLS
8430 RETURN
```

LATTICE PATTERNS

In the last instalment, we investigated a set of symmetrical patterns along a line, developing a procedure to draw these. Here, we extend our investigation of transformations to include patterns in two dimensions. To begin with, we look at how LOGO can create the grids on which these patterns are based.

If, instead of translating the motif we defined in the last article (see page 915) along a line, we allow two non-parallel translations to occur simultaneously, then our pattern becomes two-dimensional. We'll begin investigating this set of patterns by using a single dot as our unit.

```
TO DOT
   PD FD 1 BK 1 PU
END
```

The procedure to perform these simultaneous translations is defined as follows:

```
TO GRID :STARTX :STARTY :XSTEP :YSTEP :ANGLE
   DRAW HT PU
   SETXY :STARTX :STARTY SETH 0
   REPEAT 3 [LINE :XSTEP DOWN :YSTEP :ANGLE]
END
```

This procedure draws a grid of nine points. The inputs give the co-ordinates of the starting point, the size of the X and Y steps, and the heading of the corresponding points of the next row from the present row. The LINE procedure:

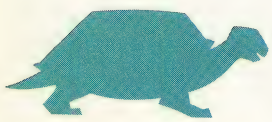
```
TO LINE :X
   REPEAT 3 [UNIT SETX XCOR + :X]
   SETX XCOR — 3 * :X
END
```

draws a single line of three units across the screen, and then returns the turtle to its starting point. For the moment, the UNIT procedure is simply a dot:

```
TO UNIT
   DOT
END
```

Another procedure:

```
TO DOWN :Y :A
   SETH :A
   FD :Y
   SETH 0
END
```



moves the turtle to the next row (in the way we have used it, this has involved moving 'down' to the next row, which gives the procedure its name), and then restores the heading. The five types of plane lattice, along with LOGO procedures to draw them, are given in the diagram.

Many different patterns can be obtained from combinations of these basic grids, although it is probably more interesting to change the procedure to draw the line at various angles rather than straight across the screen.

Another line of investigation is to see how the symmetry of each of the grids can be enhanced by giving various forms of symmetry to the unit drawn at each point. There are 17 such patterns and they are all shown in the second diagram. An obvious method of drawing out all these 17 possibilities is to replace the UNIT command in the LINE procedure with a procedure that draws the shape at that point.

THE UNIT SHAPE

The unit shapes are made from a basic motif together with various reflections and rotations. As a demonstration, let's define our basic motif, and call it LIT. (As before, this is state transparent and uses no subprocedures.)

```
TO LIT
   PD
   FD 15
   RT 90
   FD 5
   BK 5
   LT 90
   BK 15
   PU
END
```

We can use the procedures we developed in the previous article to create two procedures: MOTIF and its mirror image, R.MOTIF:

```
DEFINE "MOTIF TEXT "LIT
DEFINE "R.MOTIF REWRITE "LIT
```

We can now define the unit shapes. For example, the units for patterns 7 and 17 are:

```
TO UNIT7
   LT 90 MOTIF RT 90
END

TO UNIT17
   RT 30
   REPEAT 6 [MOTIF R.MOTIF RT 60]
   LT 30
END
```

If you look back at the LINE procedure, you will see that it runs the procedure UNIT. Therefore, in order to draw pattern 7, for example, the procedure UNIT must be changed to read as UNIT7. This we do by using DEFINE.UNIT7 where:

```
TO DEFINE.UNIT :NUMB
   DEFINE "UNIT TEXT WORD "UNIT :NUMB
END
```

We will now run the grid drawing and unit drawing parts of a pattern at the same time. A procedure called PAT lets us do this:

```
TO PAT :GRID :NUMB :PROC
   DEFINE "MOTIF TEXT :PROC
   DEFINE "R.MOTIF REWRITE :PROC
   DEFINE.UNIT :NUMB
   RUN ( LIST :GRID )
   ERASE MOTIF
   ERASE R.MOTIF
   ERASE UNIT
END
```

To draw pattern 17 we would now type:

```
PAT "HEX 17 "LIT
```

This draws a hexagonal grid, with UNIT17 at each point, using LIT as the basic motif.

This method works well for all the patterns except 4, 6, 7 and 12. In these cases, the unit shape is not the same at each point, but instead undergoes a transformation (reflection, rotation, or both together). One way of dealing with this is to incorporate these transformations into the LINE and DOWN procedures. So we'll define TRANX as the transformation to be applied to the basic translation across the screen, and TRANY will be the transformation to be applied between rows. LINE and DOWN then become:

```
TO LINE :X
   REPEAT 3 [UNIT SETX XCOR XCOR + :X TRANX]
   SETX XCOR — 3 * :X
END

TO DOWN :Y :A
   SETH :A
   FD :Y
   SETH 0
   TRANY
END
```

We now define pattern 7 as:

```
TO PATTERN7 :PROC
   DEFINE "TRANX [[] [REFLECT RT 180]]
   DEFINE "TRANY [[] []]
   PAT "RECT 7 :PROC
   ERASE TRANX
   ERASE TRANY
END
```

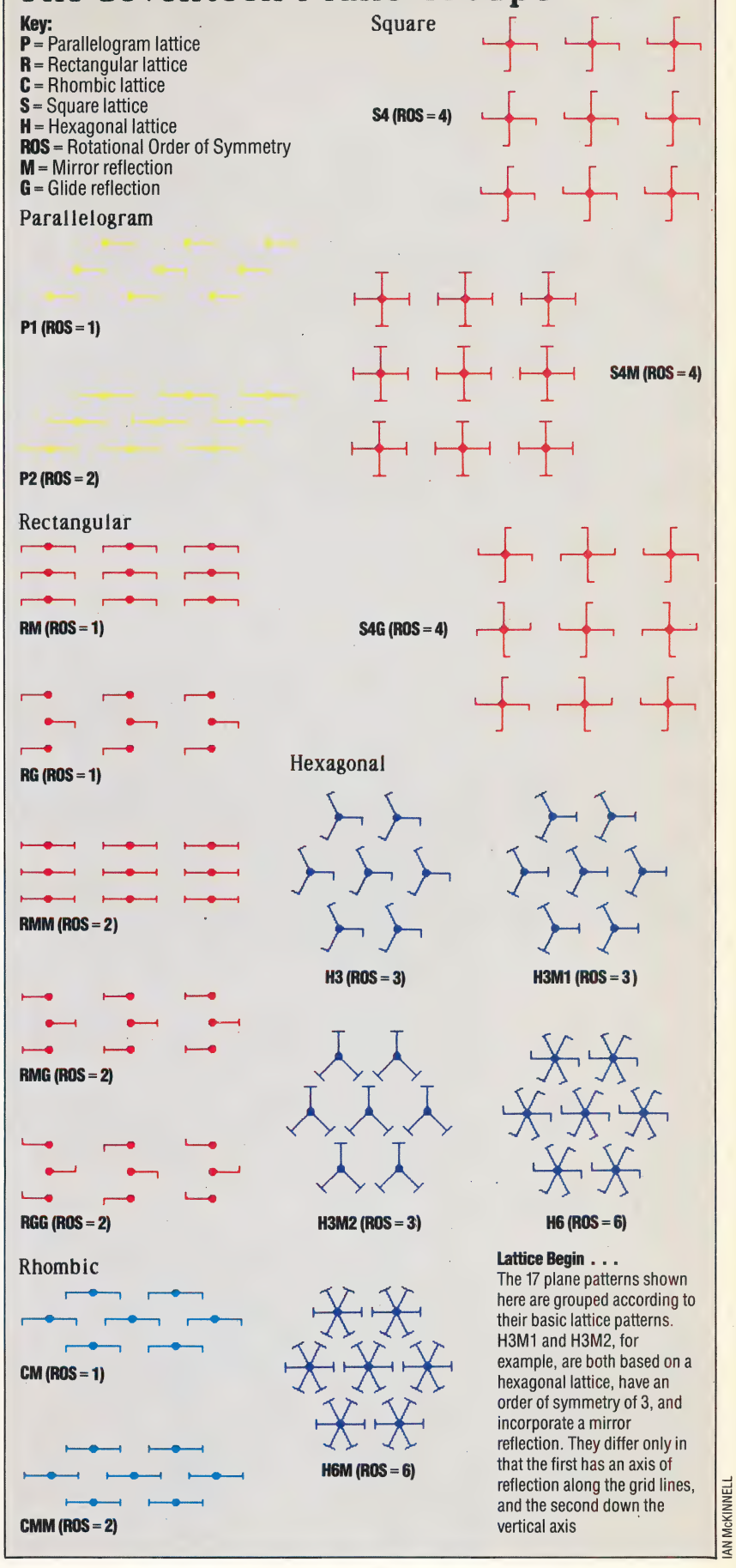
To use this, enter PATTERN7 "LEG. After running the above procedure, TRANX would have been defined as:

```
TO TRANX
   REFLECT
   RT 180
END
```

REFLECT is used to reflect the unit pattern. This procedure is defined by rewriting the UNIT

The Seventeen Plane Groups

Key:
P = Parallelogram lattice
R = Rectangular lattice
C = Rhombic lattice
S = Square lattice
H = Hexagonal lattice
ROS = Rotational Order of Symmetry
M = Mirror reflection
G = Glide reflection



Lattice Begin . . .
The 17 plane patterns shown here are grouped according to their basic lattice patterns. H3M1 and H3M2, for example, are both based on a hexagonal lattice, have an order of symmetry of 3, and incorporate a mirror reflection. They differ only in that the first has an axis of reflection along the grid lines, and the second down the vertical axis

IAN McKINNELL

The Five Types of Plane Lattice

Parallelogram

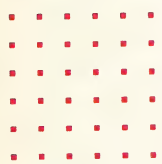
```
TO PARALLEL
  GRID ( – 60 ) 90 80 50 205
END
```

Rhombic

```
TO RHOMB
  GRID ( – 30 ) 90 80 80 225
END
```

Rectangular

```
TO RECT
  GRID ( – 80 ) 90 80 50 180
END
```



Square

```
TO SQUARE
  GRID ( – 80 ) 90 80 80 180
END
```

Hexagonal

```
TO HEX
  GRID ( – 30 ) 90 80 80 210
END
```

procedure:

```
TO REFLECT
  DEFINE "UNIT REWRITE "UNIT
END
```

The rewriting now involves replacing RT with LT, and vice versa, as well as MOTIF with R.MOTIF, and vice versa. Our previous version of the rewriting procedure swapped only RT and LT (see page 915). To modify it, all we need to do is change CHANGE.WORD — which now becomes:

```
TO CHANGE.WORD :WORD
  IF ( ANYOF :WORD = "RT :WORD = "RIGHT ) THEN
  OUTPUT "LEFT
  IF ( ANYOF :WORD = "LT :WORD = "LEFT ) THEN
  OUTPUT "RIGHT
  IF :WORD = "MOTIF THEN OUTPUT "R.MOTIF
  IF :WORD = "R.MOTIF THEN OUTPUT "MOTIF
  OUTPUT :WORD
END
```

Another way to approach this problem would be to use the version of REWRITE that also changes the subprocedures of the input procedure. We give this version amongst the answers.

For most of the patterns, the movement between points is simply a translation, and there are no other transformations to be performed. TRANX and TRANY, therefore, don't do anything. PATTERN17 is an example of this type:

```
TO PATTERN17 :PROC
  DEFINE "TRANX [[] []]
  DEFINE "TRANY [[] []]
  PAT "HEX 17 :PROC
  ERASE TRANX
  ERASE TRANY
END
```

Having covered all the basic possibilities, we leave the rest of the patterns for you to define.

Logo Flavours

For all LCSI versions:

Use CS for DRAW
Use OR for ANYOF
SETPOS, followed by a list, is used for SETXY
IF has a different syntax:

```
IF :WORD = MOTIF [OUTPUT "R.MOTIF]
```

TEXT and DEFINE do not exist as primitives in Atari LOGO, although the Atari manual does give a method of defining them

Exercise Answers

1. To rotate a shape about the point (X, Y) through an angle of A degrees:

```
TO ROTATE :X :Y :A
  PU
  MAKE "H HEADING
  MAKE "XOLD XCOR
  MAKE "YOLD YCOR
  MAKE "R SQRT ( :XOLD — :X ) * ( :XOLD —
  :X ) + ( :YOLD — :Y ) * ( :YOLD — :Y )
  PU
  SETXY :X :Y
  SETH TOWARDS :XOLD :YOLD
  RT :A
  FD :R
  SETH :H + :A
  PD
END
```

2. A rewrite procedure that will rewrite subprocedures as well.

```
MAKE "PROCS.DONE []

TO REWRITE :PROC
  MAKE "PROCS.DONE FPUT :PROC
  :PROCS.DONE
  OUTPUT REWRITE.PROC TEXT :PROC
END

TO REWRITE.PROC :TEXT
  IF :TEXT = [] THEN OUTPUT []
  OUTPUT FPUT REWRITE.LINE FIRST :TEXT
  REWRITE.PROC BUTFIRST :TEXT
END

TO REWRITE.LINE :LINE
  IF :LINE = [] THEN OUTPUT []
  IF LIST? FIRST :LINE THEN OUTPUT FPUT
  REWRITE.LINE FIRST :LINE REWRITE.LINE
  BUTFIRST :LINE
  OUTPUT FPUT CHANGE.WORD FIRST :LINE
  REWRITE.LINE BUTFIRST :LINE
END

TO CHANGE.WORD :WORD
  IF ( ANYOF :WORD = "RT :WORD = "RIGHT )
  THEN OUTPUT "LEFT
  IF ( ANYOF :WORD = "LT :WORD = "LEFT )
  THEN OUTPUT "RIGHT
  IF PROCEDURE? :WORD THEN
  SUBPROCEDURE :WORD OUTPUT WORD "£ :
  WORD
  OUTPUT :WORD
END

TO PROCEDURE? :NAME
  IF NUMBER? :NAME OUTPUT "FALSE
  IF LIST? :NAME OUTPUT "FALSE
  TEST WORD ? :NAME
  IFTRUE IF WORD? TEXT :NAME OUTPUT
  "FALSE ELSE IF NOT ( TEXT :NAME = [] )
  OUTPUT "TRUE
  OUTPUT "FALSE
END

TO SUBPROCEDURE :WORD
  IF MEMBER? :WORD :PROCS.DONE THEN
  STOP
  DEFINE ( WORD "£ :WORD ) REWRITE
  :WORD
END
```

SMOOTH MOVER

Many arcade-type computer games use a scrolling background to give a sense of rapid movement. The Commodore 64 supports 'smooth' scrolling (i.e. one pixel at a time) in both the vertical and horizontal directions. We create a routine to scroll a background design horizontally across the 64's screen.

The Commodore video controller (VIC) chip can displace the Commodore screen by up to eight pixels in either direction. Horizontal displacement is controlled by the lowest three bits of the VIC register at location 53270 ($D016). Setting these three bits to values from 7 to 0 in sequence progressively displaces the screen one pixel to the left. In BASIC, we would use the following POKE statement:

```
POKE 53270, (PEEK(53270) AND 248)+P
```

where P has a value from 0 to 7.

By combining this facility with a machine code routine that moves all screen data one cell to the left and introduces a new column of data at the right-hand edge, we can produce a smooth scrolling effect. So that data can appear to scroll smoothly into and out of view, the Commodore 64 screen width should be reduced to 38 columns, instead of the normal 40 columns. To change to 38-column mode, bit 3 of the horizontal-scrolling register should be set to zero. In BASIC, we make the following POKE:

```
POKE 53270,PEEK(53270)AND247
```

The screen can be reset to the normal 40 columns by setting bit 3 to one.

The flowchart details the various tasks that have to be carried out to produce smooth horizontal scrolling. It is important to note that if we move or insert screen data, we must also make corresponding changes to the colour data.

MOVING SCREEN DATA

In principle, this task is straightforward. The screen data is normally held in 1,000 consecutive bytes starting at location 1024 ($0400): the first 40 bytes making up the top row, the next 40 forming the second row, and so on. To make the data appear to move one place left, we simply have to move each byte of data into the byte below its original position. This section of the routine employs zero-page pointers and indirect addressing to move each byte of screen and colour data one byte lower in memory.

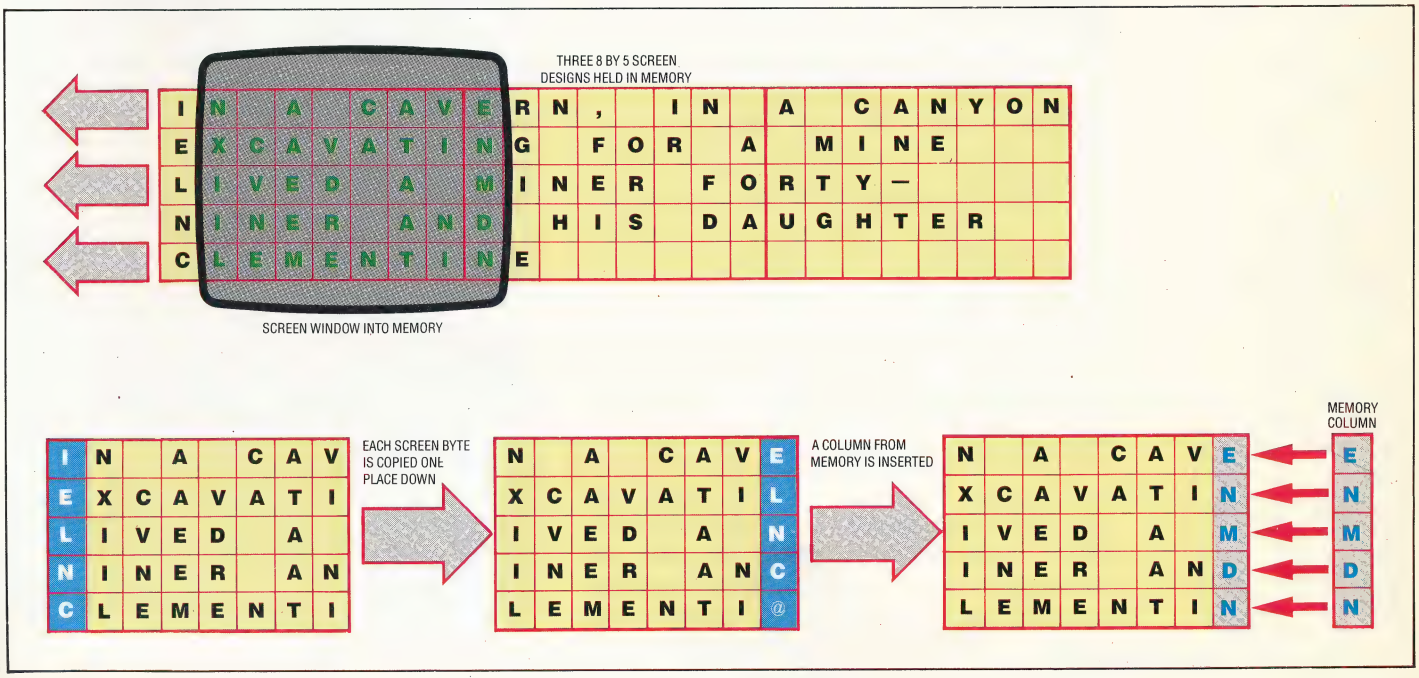
If we call the base address of the screen area SB, then the last cell in the top row will be SB+39, the last cell in the second row will be SB+79, and so on. So that the data to be scrolled onto the screen can be stored in memory in a similar way to the actual screen — that is, in packets of 1,000 bytes — the data for insertion to the right-hand edge of the screen will be the first, 41st, 81st (and so on) bytes of the area we have set side for that data. The following diagram clarifies this:

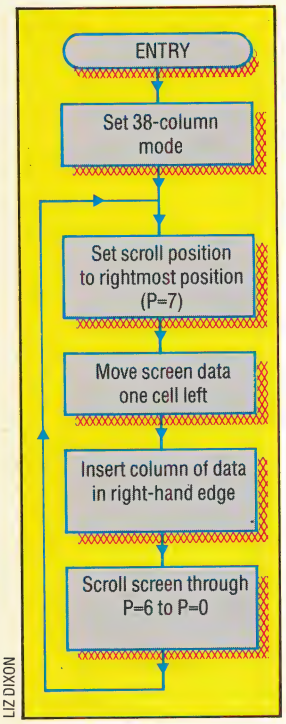
Scroll Your Own Screen

Scrolling screen designs onto the VDU from memory involves three main stages. First of all, each byte in the screen memory area is moved down one position. Because the screen is set out so that screen rows are held as sequential bytes, this has the effect of making each character on the screen appear to move one place to the left, with the exception of the characters appearing in the leftmost column of the screen. Each character in this column appears to 'wrap around' to the rightmost column. The top left screen character disappears during this process and a spurious character enters the screen area in the bottom left corner.

The second phase involves copying the relevant column from the screen memory into the rightmost column of the screen.

Having done this, the VIC chip scrolling registers can be manipulated repeatedly to make the screen appear to scroll a pixel at a time into the visible screen area





LIZ DIXON

Steps Towards A Smooth Scroll

To achieve smooth scrolling, we can use a special facility offered by the Commodore 64's video control chip — VIC — which has special scrolling registers that allow the visible screen to move from its normal position relative to the border. Single pixels in the horizontal or vertical direction can be produced. Combining this effect with character copying in machine code, we can produce smooth scrolling on a reduced 38-column screen

A pointer to the memory area is initially set to point to the byte at the beginning of the memory area to be scrolled onto the screen. Once the first column has been scrolled on, then the pointer can be incremented by one to copy a second column onto the right-hand edge of the screen, from where it can be scrolled to the left. After this process has been repeated 40 times, a complete screen of data will have been scrolled on. The memory pointer should then be increased by 960 (1000-40) to point to the beginning of the next screen.

This process must be duplicated for the corresponding area of colour data. To simplify this, we should make the address of each byte in the colour map have a constant offset to the address of the corresponding byte in the screen data map. The process can be repeated for as many screens of data as have been designed and held consecutively in memory.

In order to use the scroll routine, several pieces of information must be passed before calling it. The routine needs to know:

1) The start address of the memory area where the screen data to be scrolled is held.
2) The offset to the corresponding colour data.
3) The number of screens of data to be scrolled on.
4) A delay value, used to slow down the smooth scroll operation.

This data should be POKEd to the locations set aside in the machine code program.

Basic Calling Program

```
10 REM ***************************
20 REM ***************************
30 REM **                       **
40 REM **  BASIC CALLING PROG   **
50 REM **  FOR SCROLL ROUTINE   **
60 REM **                       **
70 REM ***************************
80 REM ***************************
90 :
95 DN=8:REM FOR CASS DN=1
100 IFA=0THENA=1:LOAD"SCROLL.HEX",DN,1
110 POKE55,0:POKE56,32:CLR:REM LOWER MEMTOP
115 REMGOSUB1000:REM SET UP SIMPLE DISPLAY
120 :
130 LMEM  =49664:    REM START OF MEMORY
140 HMEM  =49665:    REM AREA
150 LCOFF =49666:    REM OFFSET TO COLOUR
160 HCOFF =49667:    REM MAP
170 NMSCR =49668:    REM NUMBER OF SCREENS
190 DELAY =49669:    REM DELAY VALUE
200 SCROLL=49670:    REM PROG START ADDRESS
210 :
220 REM PRINTCHR$(147):REM CLEAR SCREEN
230 INPUT"DECIMAL START ADDRESS";SA
240 HS=INT(SA/256):LS=SA-HS*256
250 POKELMEM,LS:POKEHMEM,HS
260 :
270 INPUT"NUMBER OF SCREENS";NS
290 POKE NMSCR,NS
300 :
310 INPUT"DECIMAL OFFSET TO COLOUR MAP";OS
320 HO=INT(OS/256):LO=OS-HO*256
330 POKELCOFF,LO:POKEHCOFF,HO
340 :
350 INPUT"DELAY VALUE < 256";DV
360 IF DV>255 OR DV<0 THEN 350
370 POKEDELAY,DV
380 :
390 SYS SCROLL
400 POKE53270,PEEK(53270)OR8
```

The program loads the machine code into memory and asks for the information required via INPUT statements. The program splits this information into LO-byte/HI-byte form where necessary and POKEs it to the storage spaces allocated at the beginning of the machine code program. The machine code routine is then called.

Any start address, offset and number of screens may be specified, although the results will not be very meaningful if you don't put any screen designs in the memory area specified. You can test your program by loading and running the short BASIC program that sets up two simple screens of data starting at location 8192. The offset to the colour data area is 3,000 bytes. To scroll this data area onto the screen, the following information must be given in response to the prompts from the calling program:

1) Decimal start address:	8192
2) Colour offset:	3000
3) Number of screens:	2
4) Delay:	255

Basic Loader

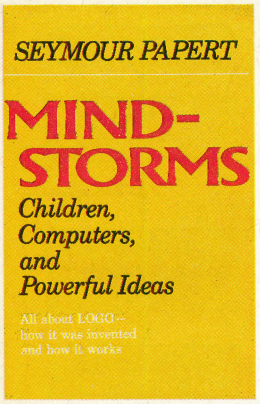
```
10 REM ***********************
15 REM **  BASIC LOADER     **
20 REM ** FOR HORIZONTAL    **
30 REM ** SCROLL ROUTINE    **
40 REM ***********************
50 :
60 FOR I=49670 TO 49945
70 READ A:POKEI,A
80 CC=CC+A
90 NEXT
92 READ CS:IF CS<>CC THEN PRINT"CHECKSUM ERROR":STOP
100 DATA173,22,208,41,247,141,22,208
110 DATA174,4,194,160,40,138,72,152,72
120 DATA173,22,208,41,248,24,105,7,141
130 DATA22,208,169,0,133,251,169,4,133
140 DATA252,169,0,133,253,169,216,133
150 DATA254,162,3,160,1,177,251,136
160 DATA145,251,200,177,253,136,145
170 DATA253,200,200,208,241,230,252
180 DATA230,254,177,251,198,252,136
190 DATA145,251,200,177,253,198,254
200 DATA136,145,253,230,252,230,254
210 DATA202,208,213,160,1,177,251,136
220 DATA145,251,200,177,253,136,145
230 DATA253,200,200,192,232,144,239
240 DATA173,0,194,133,253,173,1,194
250 DATA133,254,169,39,133,251,169,4
260 DATA133,252,32,241,194,173,0,194
270 DATA24,109,2,194,133,253,173,1,194
280 DATA109,3,194,133,254,169,39,133
290 DATA251,169,216,133,252,32,241,194
300 DATA162,6,173,22,208,41,248,141,22
310 DATA208,138,24,109,22,208,141,22
320 DATA208,172,5,194,136,208,253,202
330 DATA16,231,173,0,194,24,105,1,141
340 DATA0,194,173,1,194,105,0,141,1
350 DATA194,104,168,104,170,136,240,3
360 DATA76,19,194,173,0,194,24,105,192
370 DATA141,0,194,173,1,194,105,3,141
380 DATA1,194,202,240,3,76,17,194,96
390 DATA162,25,160,0,177,253,145,251
400 DATA202,240,29,165,251,24,105,40
410 DATA133,251,165,252,105,0,133,252
420 DATA165,253,24,105,40,133,253,165
430 DATA254,105,0,133,254,76,245,194
440 DATA96
450 DATA40227:REM*CHECKSUM*
```

Set Up Display Routine

```
1000 REM **** SET UP DISPLAY ****
1010 CL=3000:REM OFFSET TO COLOUR MAP
1020 SS=8192:REM START OF DISPLAY MAP
1030 FORI=SS TO SS+479
1040 POKEI,1:REM SCREEN CODE FOR 'A'
1050 POKEI+CL,1:REM WHITE
1060 POKEI+480,2:REM SCREEN CODE FOR 'B'
1070 POKEI+CL+480,14:REM LIGHT BLUE
1080 NEXT
1085 FORI=SS+960TOSS+999
1090 POKEI,3:REM SCREEN CODE FOR 'C'
1100 POKEI+CL,3:REM CYAN
1110 NEXT
1999 :
2020 SS=9192:REM NEXT SCREEN START
2030 FORI=SS TO SS+479
2040 POKEI,3:REM SCREEN CODE FOR 'C'
2050 POKEI+CL,5:REM GREEN
2060 POKEI+480,4:REM SCREEN CODE FOR 'D'
2070 POKEI+CL+480,0:REM BLACK
2080 NEXT
2085 FORI=SS+960TOSS+999
2090 POKEI,5:REM SCREEN CODE FOR 'E'
2100 POKEI+CL,2:REM RED
2110 NEXT
```

Horizontal Scrolling

```
;++++++++++++++++++++++++++
;++++++++++++++++++++++++++
;++                      ++
;++  HORIZONTAL SCROLL  ++
;++                      ++
;++     FOR CBM 64       ++
;++                      ++
;++++++++++++++++++++++++++
;++++++++++++++++++++++++++
;
SCRPTR=$FB              ;Ø PAGE
COLPTR=$FD              ;COPY POINTERS
;
MEMPTR=$FD              ;Ø PAGE PTR TO MEMORY
SCRLRG=$D016            ;HORIZ SCROLL REGISTER
SCRNLO=$ØØ              ;SCREEN START LOBYTE
SCRNHI=$Ø4              ;SCREEN START HIBYTE
COLRLO=$ØØ              ;COLOUR START LOBYTE
COLRHI=$D8              ;COLOUR START HIBYTE
BLOCKS=$Ø3              ;3*256 BYTE BLOCKS
EXTRA  =$E8             ;EXTRA BYTES TO 1ØØØ
NMCOLS=$28              ;NO OF COLUMNS
NMROWS=$19              ;NO OF ROWS
;
*=$C2ØØ
;
MEMLO  *=*+1            ;START OF MEMORY
MEMHI  *=*+1            ;TO BE SCROLLED
COFFLO *=*+1            ;OFFSET TO COLOUR MAP
COFFHI *=*+1
NMSCRN *=*+1            ;NUMBER OF SCREENS
DELAY  *=*+1            ;DELAY LOOP VALUE
;
;++++ SET 38 COLUMN MODE ++++
;
       LDA SCRLRG
       AND #$F7
       STA SCRLRG
;
;++++ SET 1ST SCROLL POSITION ++++
;
       LDX NMSCRN
NEXSCR
       LDY #NMCOLS
START
       TXA
       PHA              ;PUSH X,Y REGS
       TYA              ;ONTO STACK
       PHA
;
       LDA SCRLRG
       AND #$F8
       CLC
       ADC #$Ø7
       STA SCRLRG
;
;++++ COPY SCRN & COLR ONE LEFT ++++
;
       LDA #SCRNLO
       STA SCRPTR       ;SET UP Ø PAGE
       LDA #SCRNHI
       STA SCRPTR+1     ;POINTERS FOR
       LDA #COLRLO
       STA COLPTR       ;COPY
       LDA #COLRHI
       STA COLPTR+1
;
       LDX #BLOCKS
AGAIN
       LDY #$Ø1
NEXT
       LDA (SCRPTR),Y
       DEY
       STA (SCRPTR),Y
       INY
       LDA (COLPTR),Y
       DEY
       STA (COLPTR),Y
       INY
       INY
       BNE NEXT
;
;++ COPY OVER PAGE BOUNDARY ++
       INC SCRPTR+1     ;INC HIBYTES
       INC COLPTR+1     ;OF Ø PAGE PTRS
       LDA (SCRPTR),Y
       DEC SCRPTR+1
       DEY
       STA (SCRPTR),Y;COPY OVER PAGE
       INY
       LDA (COLPTR),Y
       DEC COLPTR+1
       DEY
       STA (COLPTR),Y
       INC SCRPTR+1     ;INC Ø PAGE PTRS
       INC COLPTR+1     ;AGAIN
       DEX
       BNE AGAIN
;
;++ DO EXTRA BYTES ++
       LDY #$Ø1
ANTHER
       LDA (SCRPTR),Y
       DEY
       STA (SCRPTR),Y
       INY
       LDA (COLPTR),Y
       DEY
       STA (COLPTR),Y
       INY
       INY
       CPY #EXTRA
       BCC ANTHER       ;IF Y<EXTRA REPEAT
;
;++++ INSERT RIGHT COLUMN OF SCREEN ++++
;
       LDA MEMLO
       STA MEMPTR
       LDA MEMHI        ;SET UP Ø PAGE
       STA MEMPTR+1     ;PTRS TO MEMORY
       LDA #NMCOLS-1
       STA SCRPTR       ;SET UP Ø PAGE
       LDA #SCRNHI      ;PTRS TO SCREEN
       STA SCRPTR+1
       JSR COPY4Ø       ;COPY COLUMN
;
;++++ INSERT RIGHT COLUMN OF COLOUR ++++
;
       LDA MEMLO
       CLC
       ADC COFFLO       ;ADD OFFSET TO
       STA MEMPTR       ;COLOUR MAP
       LDA MEMHI        ;AND SET Ø PAGE
       ADC COFFHI       ;PTRS
       STA MEMPTR+1
       LDA #NMCOLS-1    ;SET UP Ø PAGE
       STA SCRPTR       ;PTRS TO COLOUR
       LDA #COLRHI      ;RAM
       STA SCRPTR+1
       JSR COPY4Ø       ;DO COPY
;
;++++ SCROLL POSITIONS 6 TO Ø ++++
;
       LDX #$Ø6
MORE1
       LDA SCRLRG
       AND #$F8
       STA SCRLRG
       TXA
       CLC
       ADC SCRLRG
       STA SCRLRG
;
       LDY DELAY        ;COUNT DOWN
MORE2
       DEY              ;DELAY VALUE
       BNE MORE2
       DEX
       BPL MORE1
;
;++++ INCREMENT MEMORY POINTER ++++
;
       LDA MEMLO
       CLC
       ADC #$Ø1
       STA MEMLO
       LDA MEMHI
       ADC #$ØØ
       STA MEMHI
;
;++++ CHECK FOR END OF MEMORY AREA ++++
;
       PLA
       TAY              ;GET X,Y REGS BACK
       PLA
       TAX              ;OFF STACK
       DEY
       BEQ NOJMP
       JMP START
NOJMP
       LDA MEMLO
       CLC
       ADC #$C0         ;ADD 1ØØØ-4Ø
       STA MEMLO        ;TO MEMORY POINTER
       LDA MEMHI
       ADC #$Ø3
       STA MEMHI
       DEX
       BEQ RETRN
       JMP NEXSCR
RETRN
       RTS
;
;++++ COPY EVERY 4ØTH BYTE S/R ++++
;
COPY4Ø
       LDX #NMROWS
       LDY #$ØØ
REPEAT
       LDA (MEMPTR),Y
       STA (SCRPTR),Y
       DEX
       BEQ FINISH       ;EVERY ROW DONE?
;
       LDA SCRPTR
       CLC
       ADC #NMCOLS      ;INC PTRS BY 4Ø
       STA SCRPTR
       LDA SCRPTR+1
       ADC #$ØØ
       STA SCRPTR+1
;
       LDA MEMPTR
       CLC
       ADC #NMCOLS
       STA MEMPTR
       LDA MEMPTR+1
       ADC #$ØØ
       STA MEMPTR+1
       JMP REPEAT
FINISH
       RTS
```



Mindstorms by Seymour Papert, Harvester Press, 1980, £4.95 ISBN 0-71080-472-5

We begin an occasional series of book reviews by looking at two volumes that examine computing's inner life. Seymour Papert's book reveals how an educational philosophy, and the experience of watching children learn, led to the creation of a new computer language; Tracy Kidder describes the 'strange, half-mad beauty' of computers and the brilliant minds who invented one.

MINDSTORMS

'The Gears Of My Childhood' is Seymour Papert's title for the foreword to *Mindstorms*, in which he explains his book, his ideas and himself. His childhood fascination with the gear-trains in a construction set, and how he discovered the joys of

'A new world of personal computing is about to come into being . . . inseparable from the story of the people who will make it.'

learning through them is the story about which his swirling monologue is declaimed. He fell in love with the gears as a symbol of childhood's promise, and remains infatuated with their modern incarnation — the personal computer in the playroom.

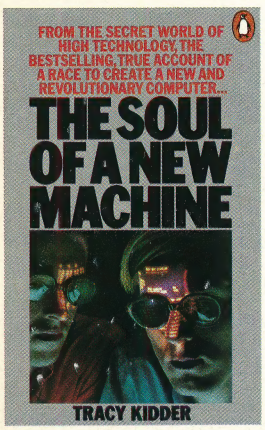
Papert's is a dominating style and personality, but the book is full of the name and the thoughts of Jean Piaget (1896-1980) the most influential educational psychologist of modern times. LOGO, the computer language developed and described

'I use the image of a Mathland — where mathematics would become a natural vocabulary — to develop my idea that the computer presence could bring the humanistic and mathematical/scientific cultures together.'

in the course of this book, is really Papert's *hommage au maitre* — an attempt to make a concrete expression of Piaget's ideas about children as 'the active builders of their own intellectual structures'. In this respect, the book isn't really about LOGO or Piaget or even Papert; its real subject is how computers can create learning spaces — 'microworlds' — in which children can learn to think and reason as happily and as richly as Papert did with his allegorical gears.

Like Piaget and his writings, Papert and LOGO have been lionised by evangelists, and then criticised by revisionists, in the space of only a few years. Piaget's ideas about the different stages of child development have been questioned for their apparent determinism; while LOGO itself is now thought by some to be useful only for teaching geometry and programming, rather than being hailed as the 'philosopher's stone'.

The title, *Mindstorms*, describes and proclaims the book. Papert writes as, no doubt, he thinks — in a marvellously stimulating mixture of cool academic analysis and white-hot mad professorship.



The Soul Of A New Machine by Tracy Kidder, Penguin, 1982, £1.95, ISBN 0-163-11433-X

THE SOUL OF A NEW MACHINE

This is the microworld from the other side of the playpen bars — Piaget's children grown up and working in computers, and observed with puzzled fascination by reporter Tracy Kidder. This is the inside story of the development of Data General's Eagle minicomputer, an undertaking completed from scratch in one astonishing year. The book won the Pulitzer prize for non-fiction in 1982, and is the subject of a film. It has eerie psychological overtones for the Piagetians, from the slightly robot-like faces of the young computer scientists on the book's cover, through the Oedipal echoes of the young computer company's efforts to outdo its giant parent, DEC, to the Svengali figure of West, the project engineer and team-leader. This

'Mechanically, monotonously, the computer . . . was telling an old familiar story — the international, materialistic fairy tale come true.'

is computing's *Moby Dick* — a compelling story, simply told, of strong personalities turning commercial pursuits into personal quests.

The book is technically a splendid read, with graphic descriptions of every stage in the design and building of the 32-bit minicomputer. Kidder was able to observe much of the project's development at first hand, and he retells the explanations for the decisions and actions taken in a lean crystal-clear prose.

The book is fundamentally about the brilliant and somewhat insular existence of the technicians whose microworld is the operating systems that they invent. Kidder is equally fascinated by their personal reactions to the developing machine, by the group's tangled subculture of loyalty, pressure

'It was a different game now. Clearly, the machine no longer belonged to its makers.'

and manipulation, and by the Byzantine company politics that determine the machine's schedule and design. The lightning-rod to all this energy is Tom West, who is pictured in the book's prologue as, literally, 'A Good Man In A Storm'. His dedicated team of young engineers work all night, inspired in part by his example, yet he cynically denies them a crucial test-machine because 'unpaid overtime is cheaper than new plant.' West might be Captain Ahab or he might be Captain America — Tracy Kidder doesn't seem to have made up his mind. But his book has thrown light on a few of the remarkable men behind the machines.

HAVE YOU ORDERED YOUR VOLUME FOUR BINDER YET?

FOLD 2

IF YOU HAVE NOT ASKED FOR THE BINDERS TO BE SENT TO YOU AUTOMATICALLY, DO SO NOW, AND ENSURE THAT YOUR COPIES OF THE HOME COMPUTER ADVANCED COURSE ARE KEPT PROPERLY BOUND AND IN GOOD CONDITION FOR YEARS TO COME.

BY TICKING THE BOX OPPOSITE, YOU WILL BE SENT BINDER NUMBER 4.

☐ PLEASE SEND ME MY VOLUME 4 BINDER NOW. I ENCLOSE A CHEQUE/POSTAL ORDER FOR £3.95 (WHICH INCLUDES POSTAGE AND PACKING).

FOLD 3

BY TICKING THE OTHER BOX *AS WELL*, YOU WILL BE SENT SUBSEQUENT BINDERS FOR YOUR COLLECTION.

☐ I WOULD ALSO LIKE TO RECEIVE FUTURE BINDERS AS THEY ARE ISSUED. I UNDERSTAND THAT I WILL RECEIVE A PAYMENT ADVICE FOR £3.95 (WHICH INCLUDES POSTAGE AND PACKING) WITH EACH BINDER. IF I AM NOT SATISFIED WITH THE BINDER, I CAN RETURN IT TO YOU, WITHIN 14 DAYS, AND OWE NOTHING.

NO STAMP NECESSARY.
JUST FOLD UP THE PAGE AS INDICATED, REMEMBERING TO ENCLOSE YOUR CHEQUE/POSTAL ORDER MADE PAYABLE TO ORBIS PUBLISHING, AND SEND TO US TODAY.

FOLD 4

FOLD 4

I enclose a cheque/postal order made payable to: Orbis Publishing Ltd. for a total of £____ which I understand includes the cost of postage and packing.

NB: Please allow 28 days for the delivery of your binders.

When you have completed the order form fill in your name and address in the space provided.

Then cut along the dotted line to detach the page, enclose your cheque/postal order, and fold the page carefully – following the instructions to complete the reply paid envelope.

NO STAMP NECESSARY.

FOLD 1

IF YOU ALREADY HAVE AN ACCOUNT NO. FOR YOUR BINDERS PLEASE FILL IT IN HERE.	

Complete the section below with one letter or figure per space.

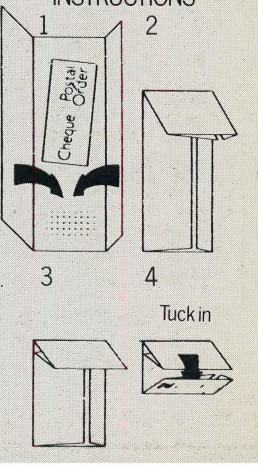
		INITIALS	SURNAME
MR			
MRS			
MISS			

ADDRESS & POST CODE

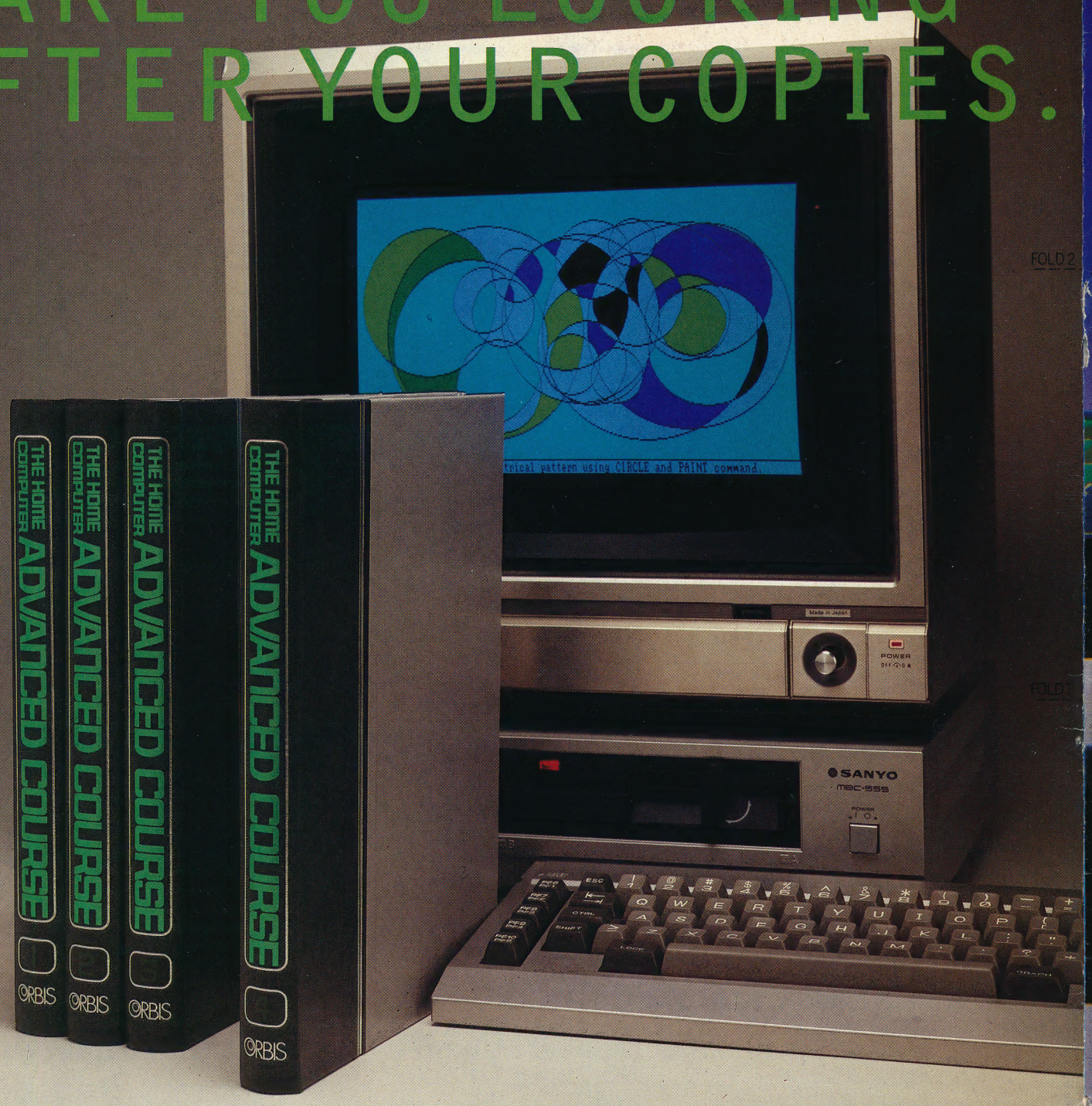
TELEPHONE NO. INCLUDING STD CODE OR EXCHANGE NAME

FOLD 1

FOLD 4
FOLDING INSTRUCTIONS



ARE YOU LOOKING AFTER YOUR COPIES.

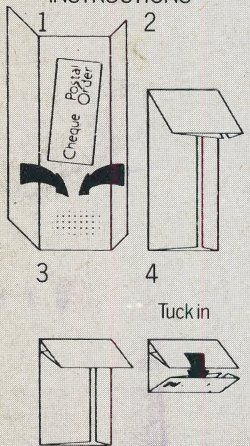


FOLD 2

FOLD 1

FOLD 4

FOLD 4
FOLDING INSTRUCTIONS



FOLD 1

Postage will be paid by licensee

Do not affix Postage Stamps if posted in Gt. Britain, Channel Islands or N. Ireland.

BUSINESS REPLY SERVICE
Licence No. SW4035.

The Home Computer
Advanced Course Binders
Orbis House, 20-22 Bedfordbury
London WC2N 4BR

|2|

FOLD 1

GUARANTEE GUARANTEE GUARANTEE GUARANTEE GUARANTEE

GUARANTEE

If you are not entirely satisfied with your binder, send it back immediately and it will be either exchanged, or, if you prefer, your money will be refunded in full.

GUARANTEE

GUARANTEE GUARANTEE GUARANTEE GUARANTEE GUARANTEE